

**Model Based Verification of the Secure Socket Layer (SSL) Protocol for
NASA Systems**

John D. Powell & David Gilliam

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Rive
M/S 125-233
Pasadena, CA 91109

Phone: 818-393-4355

Focus Issue:
System security and information assurance

Abstract

The National Aeronautics and Space Administration (NASA) has tens of thousands of networked computer systems and applications. Software Security vulnerabilities present risks such as lost or corrupted data, information theft, and unavailability of critical systems. These risks represent potentially enormous costs to NASA. The NASA Code Q research initiative "Reducing Software Security Risk (RSSR) Through an Integrated Approach" offers formal verification of information technology (IT), through the creation of a Software Security Assessment Instrument (SSAI), to address software security risks. [1,2,3,4,5,6]

The SSAI is composed of 5 parts:

1. Model Based Verification (MBV)
2. Vulnerability Matrix
3. Security Assessment Tools (SATs)
4. Property Based Testing (PBT)
5. Software Security Checklist (SSC)

This abstract discusses the verification of the Secure Socket Layer (SSL) communication protocol as a demonstration of MBV using the Flexible Modeling Framework (FMF) developed by the RSSR initiative.

MBV makes use of discrete finite models to verify critical system properties. The FMF is a generic approach to modeling and verification. However, the specific MBV and FMF properties addressed in this abstract focus on software security pertaining to the SSL protocol.

Network security properties often focus on characteristics that are manifested through the operation of multiple software components operating concurrently. The concurrent nature of the systems results in an operational space that is too large to verify testing techniques. MBV with the FMF offers

verification of critical system security properties early in the development lifecycle before an implementation exists. This makes MBV valuable because software security vulnerabilities introduced in the early lifecycle phases are costly to remove in later phases. A vulnerability that goes undetected until after system deployment results in the addition of cumbersome "patches" to mitigate the vulnerability. These "patches" may introduce new vulnerabilities in addition to mitigating the ones being corrected.

The MBV technique uses Model Checking (MC) as a core technology. Model checkers explore all paths in a finite state space from a given start state. The objective is to verify system properties over all possible system scenarios. MC provides counter examples when properties are violated, which are then used as traces for test case generation. [7,8,9]

MBV techniques, such as MC, are not without drawbacks.

- The inability to evolve a system model in a timely manner when the system definition is volatile.
- The state space explosion problem. [10] The state space that a model checker searches to verify properties grows at an exponential rate as the model becomes more detailed.

The FMF is offered as a means to bring software security issues under formal control mitigating the drawbacks of MC discussed above. The FMF seeks to achieve this by a divide and conquer approach. As such, the FMF is a: 1) System for building models in a component based manner to cope with system evolution in a timely manner, 2) Compositional verification approach to delay the effects of state space explosion for larger system models.

Modeling in a component-based manner involves the building of a series of small sub-models. Then, these components can be combined and verified over system properties of interest in a compositional manner.

The compositional verification approach used in the FMF seeks to verify properties over individual model components and then over strategic combinations of them. The goals of this approach are to:

- Infer verification results over systems that are otherwise too large for MC from results of strategic sub-model combinations.
- Retain verification results from individual components and component combinations to increase the efficiency of subsequent verifications.

Figure 1 shows how SSL model components can be mixed and matched within the FMF to verify correctness properties over multiple variations of SSL behavior. Development of a single model containing all possible behaviors can be counter-productive. Combining

behaviors that do not reasonably co-exist in a system produces many false property violations. These will flood the analyst with so much data to review that the timeliness of verification results will be compromised. Further, upon finding a valid violation of a system property the counterexample will often be convoluted by irrelevant interim model transitions. Isolating and recommending corrective action becomes a long and tedious analysis task. When the model is separated into variations through the use of FMF, valid verification knowledge can be easily extracted from the pattern of (non-) violations over the model variations. The FMF approach is a means for determining critical system functionality with regard to software security properties thereby isolating vulnerable areas for corrective actions. Finally, in an open system, such as the SSL protocol and its environment, an all encompassing model will unduly stress the limits of the test platform's memory constraints due to excessive state space explosion without the use of the FMF.

Four SSL correctness properties

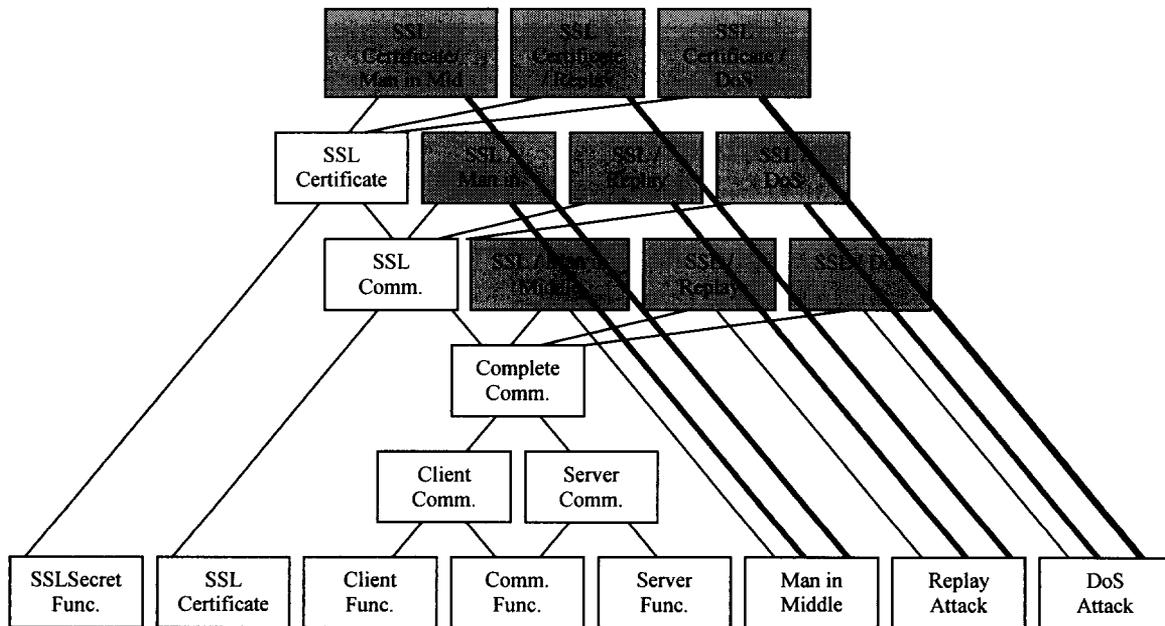


Figure 1: SSL Communication Protocol Model Component

| Each communicating entity will eventually achieve and execute the exchange of secure communication | No Attack | Man in the Middle Attack | Replay Attack | DoS attack |
|---|------------------|---------------------------------|----------------------|-------------------|
| Signed SSL Entities (Certificates) | No Violation | Violation | Violation | Violation |
| Unsigned SSL Entities (No Certificates) | No Violation | No Violation | Violation | Violation |
| Non-SSL Client Server Entities | No Violation | No Violation | Violation | Violation |

Table 1: Verification Results Summary

were verified over the FMF model components. They are 1) The SSL secure communication shall initialize eventually unless less an attack has successfully inserted itself in such a manner that the resulting secure communication will be compromised, 2) Once secure communication is established secure contacts and responses will always be reached, 3) A secure message that has been intercepted shall be detected and not accepted by the SSL recipient of the secure message, 4) Under the rules for attacks, an attack may only read unsecured messages or secured messages if the SSL secret has previously been captured. 5) Securely communicating entities shall not reveal their secret even during the handshake initialization

The significant verification results shown in Table 1 indicate that:

1. SSL entities communicate correctly when no attack is present
2. Only the SSL entities using signed certificates recognized a Man-in-the-Middle Attack, aborting before exposing secure communication.
3. The Replay Attack failed to access secure communication
4. The DoS Attack did deadlock the system but did not capture secure communication

Acknowledgement

The work described in this abstract was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

- [1] D. Gilliam, et.al., "Reducing Software Security Risk Through an Integrated Approach," Proc. of the Ninth IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (June, 2000), Gaithersburg, MD, pp.141-146.
- [2] G. Fink, M. Bishop, "Property Based Testing: A New Approach to Testing for Assurance," ACM SIGSOFT Software Engineering Notes 22(4) Jul 1997.
- [3] M. Bishop, "Vulnerabilities Analysis," Proceedings of the Recent Advances in Intrusion Detection (Sep. 1999).
- [4] J. Dodson, "Specification and Classification of Generic Security Flaws for the Tester's Assistant Library," M.S. Thesis, Dept. of Computer Science, Univ. of California at Davis, Davis CA (June 1996).
- [5] D. Gilliam, et. al., "Development of a Software Security Assessment Instrument to Reduce Software Security Risk" Proc. of the 10th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Boston, MA, pp144-149.
- [6] D. Gilliam, et. al., "Reducing Software Security Risk Through an Integrated Approach", IEEE Goddard 26th Software Engineering Workshop
- [7] W. Wen and F Mizoguchi. Model checking Security Protocols: A Case Study Using SPIN, IMC Technical Report, November, 1998.
- [8] J. Callahan, et. al. "Generating Test Oracles via Model Checking," NASA/WVU Software Research Lab, Fairmont, WV, Tech. Rpt #NASA-IVV-98-15.
- [9] P. E. Ammann, P. E. Black and W. Majurski. "Using Model Checking to Generate Test Specifications," 2nd International Conference on Formal Engineering Methods (1998) pp. 46-54.
- [10] G. Holzmann. Design and Validation of Computer Protocols. Prentice Hall 1990; ISBN: 0135399254 .