

Model Checking for Software Security Properties

John D. Powell

John.Powell@jpl.nasa.gov

Caltech, Jet Propulsion Laboratory

Abstract

This paper describes the use of the Flexible modeling Framework (FMF) for Model Checking (MC) to perform verification and search for vulnerabilities in the Secure Socket Layer (SSL) communication protocol. The wide use of SSL makes the existence of potential vulnerabilities in the protocol an extremely dangerous prospect. Therefore, the use of formal methods such as MC represents a rigorous form of technology in an environment where the discovery of problems carries with it high payoff in terms of the potential volume of security breaches. This paper will describe MC and FMF and go on to show FMF's utility in making tradeoffs precipitated by model fidelity needs, state space explosion and specific system property verification needs.

1. Introduction

Concurrent software systems and software applications are frequently subject to software security vulnerabilities that may render an otherwise secure networked software environment unsafe. The addition of software systems/ applications to an otherwise secure environment can affect the security and safety of the whole environment either by exploitable security vulnerabilities in the additional software or between two sets of individually secure software in a networked computing environment that, through their interaction, may cause an unwanted security exposure or vulnerability. Therefore, any system with such exposures or vulnerabilities can be compromised when the software on it contains insecurities or unexpectedly interacts in insecure ways. Further, any connected systems are also put at risk along with their resources, services, and data. If an intrusion goes undetected, due to the exposure or vulnerability, networked systems are subject to the domino effect, where access to one system will eventually yield access to other systems. A trusted

system on a corporate network that becomes compromised could give hackers access to critical corporate resources.

Given the conditions discussed above and the potentially catastrophic nature of intrusions into systems now and in the future, it is crucial that such vulnerabilities and unwanted exposures be identified and mitigated. Several factors cause these weaknesses. Generally, they can be traced to inadequate requirements, poor software design and development practices, mis-configurations, new modes of network attacks, and insecure interaction between systems.

Security weaknesses due to deficiencies in the software lifecycle can be mitigated through formal software assessment methodologies. Methodologies, such as model checking (MC), can provide greater assurance that software executing on critical systems and systems linked to them do not expose critical data and functional vulnerabilities resulting from inadequately specified software requirements and designs or exposures due to complex integration with other software.

This paper will present the formal modeling portion of the "Reducing Software Security Risk" research project. A new MC approach called the Flexible Modeling Framework (FMF) is part of a software security assessment instrument to assist developers in the verification of security properties in software during the early phases of the development and maintenance lifecycles. [1,2,3,4,7,8,9,10]

Modeling requirements and early lifecycle designs to discover software security vulnerabilities precipitated by the interaction of software components under development in a new system or proposed as additions to an existing system or environment provides early insight into potential software security problems. This early detection assists software development efforts to address and correct vulnerabilities at significantly less cost in terms of time and effort than allowing them to persist into later lifecycle phases. Vulnerabilities that do survive to later lifecycle phases are often addressed with

cumbersome “patches” that can introduce new security problems or yet unknown exposures or vulnerabilities of their own.

While this paper focuses on a formal modeling technique, the overall research effort has several foci and attempts to address an end-to-end software lifecycle process to reduce these unwanted exposures and vulnerabilities. Information about the overall research effort regarding network security is available at: <http://security.jpl.nasa.gov/rssr>.

Section 2 gives a brief overview of MC and the FMF rationale. Section 3 describes the use, to date, of FMF modeling techniques during verification of security properties for the SSL protocol. Finally, section 4 offers a summary conclusion of the benefits of FMF and MC during the SSL verification effort.

2. Model Based Verification through Model Checking

Model Based Verification (MBV) involves development of high fidelity abstractions (Models) of system behavior in a form that can be analyzed to determine if critical system properties hold over the specified possible behavior scenarios. MC is the specific type of MBV that is utilized in the approach described in this paper for the purpose of verifying software security properties for networked system.

Software model checkers automatically explore all paths from a start state in a computational tree that is specified in an MC model. The computational tree may contain repeated copies of sub-trees. State of the art Model Checkers such as SPIN exploit this characteristic to improve automated verification efficiency. The objective is to verify system properties with respect to models over as many scenarios as feasible. Since the models are a selective representation of functional capabilities under analysis, the number of feasible scenarios is much larger than the set that can be checked during testing. Model Checkers differ from traditional formal techniques by the following characteristics:

- Model checkers are operational as opposed to deductive
- Model checkers provide counter examples when properties are violated (error traces)
- Their goal is oriented toward finding errors as opposed to proving correctness since the model is correct.

An innovative verification approach, which employs MC as its core technology, is offered as a means to bring software security issues under formal control early in the life cycle. [11,12] The FMF seeks to address the problem of formal verification of larger systems by a divide and conquer approach. [13] First, by verifying a property over portions of the system, then incrementally inferring the results over larger subsets of the entire system. As such, the FMF is: 1) a system for building models in a component based manner to cope with system evolution over time and, 2) an approach of compositional verification to delay the effects of state space explosion. This methodology allows property verification results of large and complex models to be examined and extrapolated appropriately.

An innovative verification approach that employs MC as its core technology is offered as a means to bring software security issues under formal control early in the life cycle while mitigating the drawbacks discussed above. The FMF verifies a property over portions of the system, and then incrementally infers the results over larger subsets of the entire system. Thus, the FMF is a system for building models in a component-based manner to cope with system evolution in a timely manner.

The compositional verification approach delays the effects of state space explosion and allows property verification results to be examined and extrapolated with respect to larger, complex models. (See Figure 1)

Modeling in a component-based manner involves building a series of small models, which later will be strategically combined for system verification purposes. This strategic combination correlates the modeling function with modern software engineering and architecture practices whereby a system is divided into

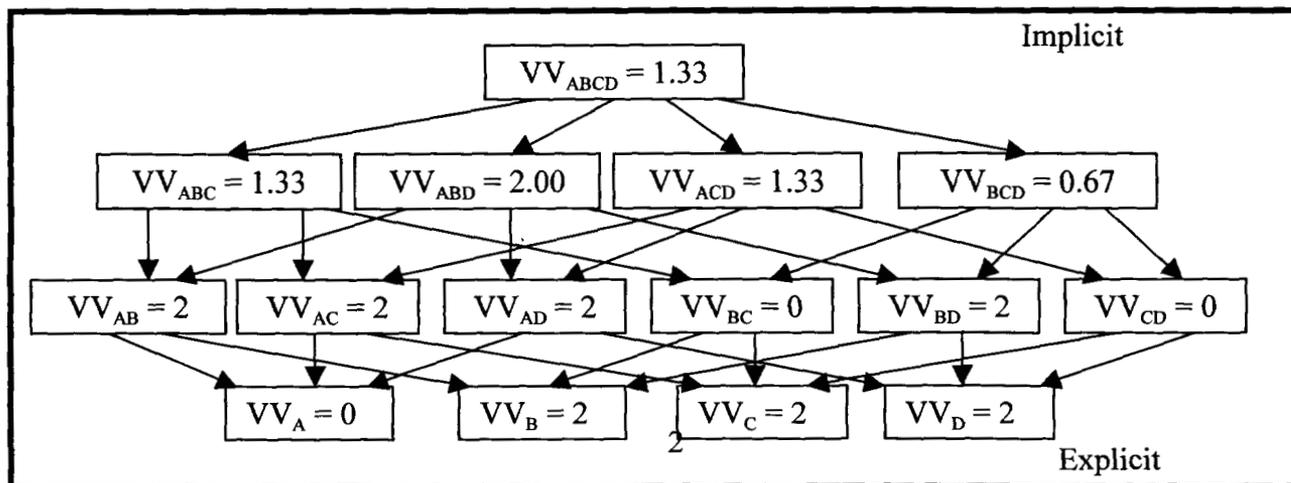


Figure 1: MCCT Verification Value Assignment and Propagation

major parts, and subsequently into smaller detailed parts, and then integrated to build up a software system. An initial series of simple components can be built when few operational specifics are known about the system. However, these components can be combined and verified for consistency with properties of interest such as software security properties.

The approach of compositional verification 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: 1) infer verification results over systems that are otherwise too large and complex for MC from results of strategic subsets (combinations) while minimizing false reports of defects; 2) retain verification results from individual components and component combinations to increase the efficiency of subsequent verification attempts in light of modifications to a component.

3. Component based MC of SSL

The current MBV efforts in the RSSR project are focused on verifying the Secure Socket Layer (SSL) protocol through the use of component based MC. As the name suggests SSL consists of layers of functionality that lend themselves to decomposition during the specification (modeling) phase and an FMF compositional approach during verification. The abstract model layers of the SSL protocol consist of:

- The SSL Engine itself
- The SSL library, which is an API over the actual engine functionality
- The application that invokes the library functions

There also exists functionality at a higher level, consisting of an application interacting with the SSL specific functionality, which can be modeled in components for further verification. (See Figure 2) While FMF techniques are used to model and verify the SSL layers in search of vulnerabilities, the FMF techniques become essential when the functionality competing and interacting with SSL functionality is added to the state space.

The process of MC, and thus FMF, involves necessary tradeoffs between desired details of the system's behaviors and the need to manage (limit) the state space explosion properties. The first phase of modeling the SSL protocol focuses on the latter issue. Therefore, the initial model components include only the most basic functional behaviors of the SSL engine. The SSL engine functionality is first modeled as binary success/failure responses with the SSL library functionality being modeled as sequences of SSL engine

actions. Then SSL application functionality can be modeled as a non-deterministic set of scenarios constrained only by the rules of the SSL specification. The verification of properties over this abstraction (model) produces little in the way of interesting verification results due to the lack of detail, and thus, interaction between model elements at the lowest levels (SSL engine). This indicates the need to increase the detail of some behaviors at the lowest levels of the system (SSL engine) to balance needed system behavior with state space explosion.

Fully specifying the details of the SSL engine will cause a state space explosion that will overwhelm computing resources during property verification. Therefore, the use of FMF is used to add detail at the lowest levels, determine MC resource thresholds, and lower detail in alternate areas through the introduction of simple non-determinism and elimination of constraining variables. While the new state-space/resource threshold, resulting from the introduction of non-determinism, allows for MC of more components in combination, the ambiguity in the models representation of system behavior resulted in an unacceptable rate of false anomaly detection. Again a trade of involving a limited increase in states space to obtain higher system fidelity by eliminating MC's ability to explore scenario paths that is impossible in the actual SSL protocol. Therefore, non-determinism was replaced with strategic use of *homomorphic reduction* as a compromise between excess detail and strict non-determinism. Homomorphic reduction is a means of controlling state space explosion by 1) identifying critical values for a variable, within the full range of values it may take on, that affect the decision gates in the logic of the rest of the system, 2) replacing the full range of values with a smaller range of values that represent transition from one sub-range of original

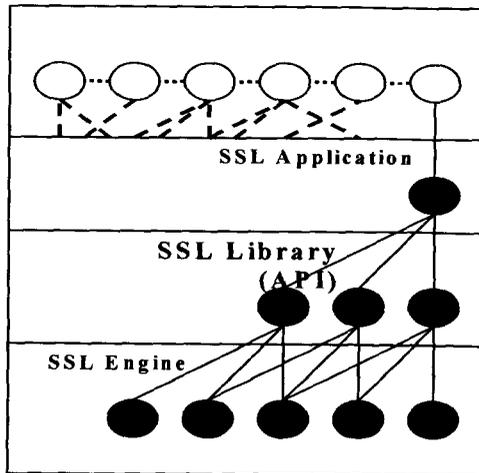


Figure 2: Component Modeling of SSL

values to another and 3) adjusting the logic in the rest of the system model to reflect the reduction. For example, instead of allowing the SSL engine function and SSL application to provide/refuse identity verification credentials non-deterministically the model takes into account a small set of binary condition that are set to true when certain prerequisite functionality has been successfully exercised. Then, the binary variables are evaluated to determine if identity verification should be granted/refused.

The FMF allows the analyst to make these changes quickly and efficiently and reverse the changes easily when the formal specification (model) is organized in a component-based manner that adheres to the FMF. As of the writing of this paper MC and the FMF have not directly uncovered any new vulnerabilities inherent in the SSL protocol. However, the process of formal specification of the protocol and exploration of potential anomalies, later found to be false, has aided security professionals at JPL in reasoning about different dimensions of the overall SSL application and its interaction with various systems at large. This has raised awareness to potential vulnerabilities at the level where the full SSL application(s) would be interacting with other applications in the networked environment despite the fact that MC with the FMF did not give direct evidence of their existence. Direct MC evidence was not possible due to state space explosion and abstraction fidelity constraints when trying to model and verify an entire networked environment of applications at large.

4. Conclusion

The FMF approach to MC has shown itself to be valuable in making tradeoffs between necessary system details and control of state space explosion for verification of system properties in a timely manner. FMF's component based approach yield models that allow for quick tradeoffs of greater detail on one portion of a model for greater abstraction in other part as a means of quickly adapting an MC model to the specific needs of different properties to be verified over it. The layered nature of the SSL protocol and digital communication ad security at large readily lends itself FMF practices.

5. Acknowledgement

The research described in this paper is being carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

6. References

- [1] D. Gilliam, J. Kelly, M. Bishop, "Reducing Software Security Risk Through an Integrated Approach," Proc. of the Ninth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (June, 2000), Gaithersburg, MD, pp.141-146.
- [2] Published and maintained by Mitre. The CVE listing can be found at: <http://cve.mitre.org/>
- [3] G. Fink, M. Bishop, "Property Based Testing: A New Approach to Testing for Assurance," ACM SIGSOFT Software Engineering Notes 22(4) (July 1997).
- [4] M. Bishop, "Vulnerabilities Analysis," Proceedings of the Recent Advances in Intrusion Detection (Sep. 1999).
- [5] J. Dodson, "Specification and Classification of Generic Security Flaws for the Tester's Assistant Library," M.S. Thesis, Department of Computer Science, University of California at Davis, Davis CA (June 1996).
- [6] J. R. Callahan, S. M. Easterbrook and T. L. Montgomery, "Generating Test Oracles via Model Checking," NASA/WVU Software Research Lab, Fairmont, WV, Technical Report # NASA-IVV-98-015, 1998.
- [7] 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.
- [8] G. Lowe. Breaking and Fixing the Needham-Schroeder Public Key Protocol Using CSP and FDR. In TACAS96, 1996.
- [9] W. Wen and F Mizoguchi. Model checking Security Protocols: A Case Study Using SPIN, IMC Technical Report, November, 1998.
- [10] G. Holzmann. Design and Validation of Computer Protocols. Prentice Hall 1990; ISBN: 0135399254 .
- [11] D. Gilliam, J. Kelly, J. Powell, M. Bishop, "Development of a Software Security Assessment Instrument to Reduce Software Security Risk" Proc. of the Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Boston, MA, pp 144-149.
- [12] D. Gilliam, J. Powell, J. Kelly, M. Bishop, "Reducing Software Security Risk Through an Integrated Approach", IEEE Goddard 26th Annual Software Engineering Workshop.
- [13] Component Based Model Checking, J. Powell, D. Gilliam, Proceedings of the 6th World Conference on Integrated Design and Process Technology, June 23-28, Pasadena CA, p66 & CD

7. Biographies

John D. Powell holds a M.S. in Computer Science from West Virginia University and is a software quality assurance researcher at the California Institute of Technology's Jet Propulsion Laboratory (JPL) in the Quality Assurance office. Currently he performs research in the area of Quality/Cost Estimation and

Prediction as well as Formal Methods research for efforts at JPL. Prior to his work at JPL/USC, John worked as a System Software IV&V Analyst for NASA's prime IV&V contractor (Titan-Averstar) performing IV&V analysis on the Redundancy Management and Control systems for the Space Shuttle's Checkout Launch and Control System (CLCS). Prior to that, at the NASA Goddard IV&V Facility, John performed research under the Intelligent Systems Initiative exploring alternatives to traditional model checking in conjunction with West Virginia University's Software Research Laboratory (SRL). His publications include a master thesis, papers at ICSE, CSEE&T, ISRE, IEEE WETICE, ISPA and various NASA Conferences/Workshops and technical reports.