Security System Property Decomposition for Model Checking with the Flexible Modeling Framework (FMF)

John D. Powell
Jet Propulsion Laboratory, California Institute of Technology
John.Powell@jpl.nasa.gov

Abstract

This paper discusses proposed improvements to the Flexible Modeling Framework (FMF) approach to model checking. These improvements, Property Decomposition and Property Models, focus on the expression of linear temporal logic properties in a conjunctive normal form and their application, in whole or in part, to subsets of systems to control state space explosion. This approach may extend the current capabilities of the FMF to allow for abstract verification of significantly large, potentially real world, system in their entirety. FMF is a method that starts at the leaf node level of the Model Component Combination Tree (MCCT), which represents individual model components and progresses "upwards" through increasingly large model component combination toward the root node (full system model). During this progression model checking becomes infeasible at some threshold of model component combination. At this point statistical propagation attempts to infer system-wide verification results with decreased confidence. Property decomposition is an alternative to statistical propagation that substitutes verification results for portions of the model checking state space. This substitution allows larger model component combinations, potentially the entire system, to be model checked.

1. Introduction

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 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 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]

MC with the FMF defers the state space explosion problem inherent in the use of model checking for verification and allows for greater flexibility in model development and adaptation. However, the FMF must rely heavily on probabilistic confidence ratings that degrade rapidly when propagating partial MC results over the entire system model. The potential for decomposing typical software security properties can aid in the ability to approach full FMF model checking of real world systems. MC results substitution, along with the development of P*roperty Models* (PMs), may aid FMF model checking in its ability to verify properties over full system models with a dramatic increase in the probabilistic confidence of its results. A property model is a high level model that takes into consideration the actual property being verified and the model components from the FMF to leverage substitution of partial verification results into the FMF.

2. MC with the FMF

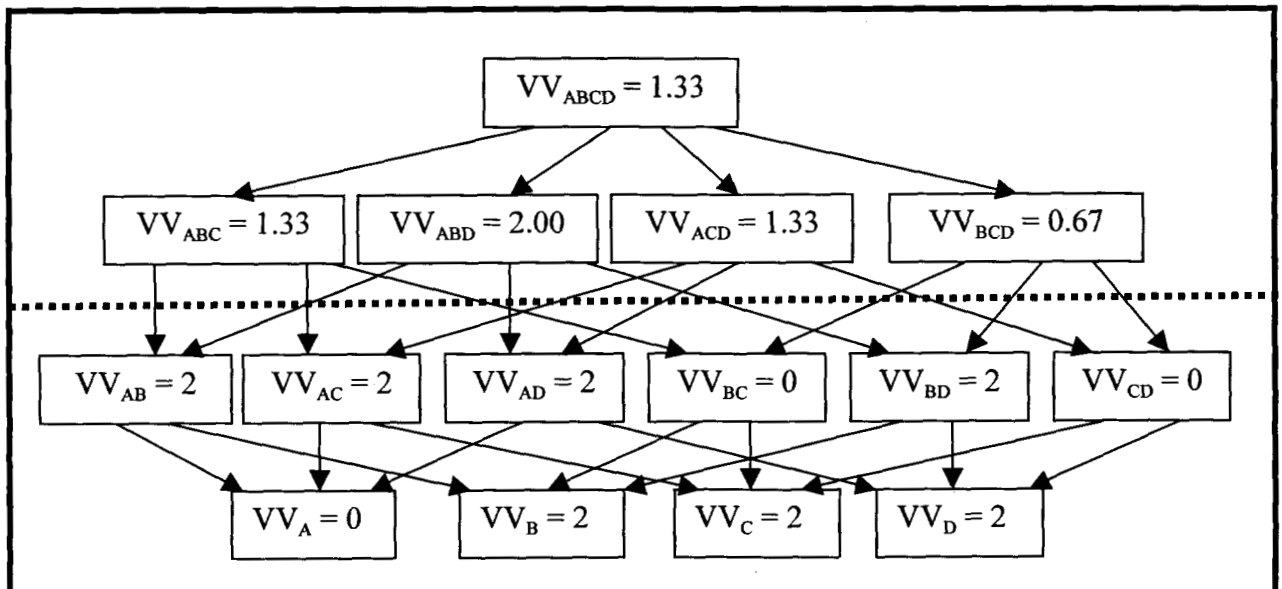

Figure 1: Model Component Combination Tree

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 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. Property Decomposition and Verification

The FMF makes use of a Model Component Combination Tree (MCCT) to propagate partial MC results up to the full system model (root node) by first MC individual components (leaf nodes) and successive combinations as it moves up the tree. A threshold is reached where the component combination is too large for MC. To date propagation from this threshold to the full system has been done by considering partial system verification results as inputs to statistical probability strategies for correct propagation. However, the confidence rations of results from these methods degrade very quickly. Property decomposition seeks to substitute the partial system (sub-tree in the MCCT) verification results with the binary state of the property being verified and/or its largest set of elements from the CNF. (See Section 3.1) This in effect reduces the larger state space represented by the MCCT sub-tree to the smallest possible state space for the particular property being verified.

Decomposition of properties for use in the FMF approach to MC consists of:

- Forming properties in a Conjunctive Normal Form (CNF) with Linear Temporal Logic (LTL) operators properly distributed throughout.
- Performing FMF model checking over the components, and their combinations, of a system model
- Replacing model component combinations with (partial) MC results.
- Continuing FMF model checking using the remaining components (combinations) and the existing MC result substitutions.
- Repeating the substitution and MC procedure until the entire system model can be check via a Property Model (PM) that reflects the system behavior only in terms of the specific variables that pertain to the property being verified.

This process transforms a general MC model of the system, and its sub-components that are too large to be model checked in combination to a property specific set of model component combinations that can be further combined while considering both system behavior and the explicit property being verified. This property specific set of model component combinations is referred to as a PM because it preserves the known behaviors of the system in the smallest possible state space terms as they pertain to the specific property in question. This continual reduction in state space size for continued MC over the FMF's MCCT may allow full MC via the FMF of very large systems with dramatically higher confidence ratings than offered by previous statistical propagation strategies.

3.1. Software Security Properties in CNF

Due to the nature of software security a great majority of the properties of interest are of the forms:

- *Always* (p *and* q *and* r *and...*)
- *Eventually* (p *and* q *and* r *and...*)

These forms consist of a CNF prepositional logic statement in the scope of a temporal operator (Always and Eventually) LTL allows the distribution of the temporal operator across the operands of a CNF statement:

- *Always*(p) *and Always*(q) *and Always* (r) *and* ...)
- *Eventually* (p) *and Eventually*(q) *and Eventually* (r) *and* ...)

The FMF Decomposition of Properties strategy leverages this CNF form of security properties. As components or component combinations go through MC and the MC threshold in the MCCT is reached the substitution of the property elements (conjunctive operands) can be substituted for a significant portion of system behavior. The implicit conjunctive relationship may be assumed. All nodes in the MCCT identified as being at the MC threshold, where no larger model component combination can be model checked, are replaced with the smaller set of property specific element states (binary). Because the individual element states are known to be related in CNF, the nodes at the MC threshold in effect may now be considered small leaf nodes in the MCCT and FMF model checking may resume. Continually repeating this process will eventually yield a single node (root node) representing the entire system whose element states can be trivially evaluated for property conformance.

6. Conclusion

The proposed improvement to the FMF approach to MC curbs state space explosion by replacing portions of the system that make large contributions to the state space with property specific equivalent spaces that are much smaller. Combining the results of these substitution of more components or other substitutions for MC yields results of larger portions of the system. The recursive application of this technique to FMF models of real world systems shows great promise for extending FMF model checking to provide high confidence property verification results for the entire system.

7. Acknowledgements

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.

8. 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