

Initial Studies of Distributed, Adaptive Program Transformation Controls

- short

Anteneh Addis Anteneh, Mike Jochen, and Lori L. Pollock
University of Delaware
email: {anteneh,jochen,pollock}@cis.udel.edu

Lisa Marvel
U.S. Army Research Laboratory
email: marvel@arl.army.mil

It has been shown that dynamically evolving mobile software can enable improved program performance [1]. These performance optimizations are particularly advantageous when programs are transformed during execution based on current input, state, and environment. Allowing mobile programs to dynamically evolve or modify as they pass through a network of computation hosts can provide increased performance gains by distributing the responsibilities for the transformation process to multiple hosts. However, allowing mobile code to transform at different nodes in a network introduces security issues associated with mobile programs.

Traditional program validation and authentication methods such as checksums and digital signatures do not adequately meet the security needs of dynamically evolving mobile programs. These validation methods are not capable of distinguishing between permissible and disallowed changes to a program. Any change to a program after the signature or checksum has been computed renders the validation data inadequate and thus these techniques become ineffective. Therefore, there exists a need for new security measures that enable users to reap the benefits of dynamically evolving mobile code while mitigating the risks of the use of these mobile programs.

In this paper, we present the initial steps towards a method to efficiently control/restrict how a program transforms on a group of networked hosts. One example environment for such a framework would be a group of nodes interconnected via a wired or wireless network operating within a potentially malicious environment. Our envisioned network environment will contain a trusted server, one or more client nodes, a group of server pool nodes, and perhaps one or more malicious nodes that may attempt to modify a program in a nefarious manner. The trusted server will distribute the original version of the program and define the transformation control policy. A client node is a node considering execution of the software and possibly requesting transformations. The server pool nodes are an odd number of hosts (at least 3) designed to assist the program transformation process.

The proposed framework will utilize specification languages to control program transformations [3]. The framework will allow the client nodes in a network to refer to a transformation control policy before accepting any proposed changes to a program. We have developed a language to handle communication of the control and requests of program transformations. Each type of transformation/optimization that we want to control must be defined in the program trans-

formation control policy. Initial transformations that the system is designed to handle include constant propagation, copy propagation, method inlining, dead code elimination, and scalar replacement of array elements and pointer accesses.

We are using Jikes RVM [2] as the framework base. Jikes RVM is a research virtual machine that performs dynamic, adaptive optimization on Java programs. Our goal is to restrict the optimizations that Jikes applies with the use of the transformation control specifications. Initial simulations indicate that we are able to control transformations applied to a program. We are currently setting up to evaluate the cost of transformations under different scenarios to measure the efficiency of our framework. We plan to show results from an evaluation of an example scenario. This evaluation will study a scenario in which (1) the client nodes of a network execute a program, (2) the client nodes submit transformation requests to the server pool, (3) the server pool validates the transformation requests, performs the validated transformations, and distributes the new code to the client nodes. Cost will be measured in terms of power, space (memory), and time.

References

- [1] Matthew Arnold, Stephen Fink, David Grove, Michael Hind, and Peter F. Sweeney. Adaptive optimization in the Jalapeño JVM. In *Proceedings of the 15th ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications*, pages 47–65. ACM Press, 2000.
- [2] Matthew Arnold, Michael Hind, and Barbara G. Ryder. Online feedback-directed optimization of Java. In *Proceedings of the 17th ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications*, pages 111–129. ACM Press, 2002.
- [3] Mike Jochen, Anteneh Addis Anteneh, Lisa M. Marvel, and Lori L. Pollock. Towards the safe use of dynamically transformed itinerant software. In *Proceedings – IEEE Military Communications Conference MILCOM*. IEEE, October 2005. In submission.