Eliminating Software Vulnerabilities Using Hyperion and Rigorous Static Analysis Methods

UT-B IDs 201403238, 201403241, 201403244, 201403252

Technology Summary

Modern society is dependent on software systems for virtually every aspect of life. However, most software is so complex that detection of errors or malicious codes is extremely difficult. Current detection techniques use functional testing of software, which alone is insufficient to catch all vulnerabilities. Hence, there is a need for rigorous static analysis methods that consider the entire input space of software.

To detect vulnerabilities, ORNL has developed a software program, Hyperion, that can "look inside" an executable program and determine a software's function or "behavior" without using the software's source code. The key is ORNL's Function Extraction (FX) technology, which directly computes the behavior of software binaries, no matter how they were originally coded. To visualize function call graphs of programs being analyzed, Hyperion uses the ORNL-developed dynamic function call graph visualization. This interactive visualization provides a graphical view for a function call graph that is comprehensive for the entire call graph it represents and allows the relations between functions to be easily explored and analyzed. Behavior Specification Units (BSUs), also developed at ORNL, recognize specific program behaviors by identifying patterns in the external function call behavior and abstract program behavior to a higher level of abstraction that nonpractitioners can understand. BSUs also enable Hyperion to perform automated classification of program behavior. Using a novel C/C++ standards-based compiler agnostic system, another ORNL invention, Hyperion can automatically find the address of main(), the starting point of a C/C++ program's unique functionality. This system computes the address of the main() function, identifies where the interesting functionality of a program begins and ends, and detects malicious software by analyzing program behavior.

Advantages

- Operates on compiled software–source code is not necessary
- Generates output in terms of external event traces–library calls, interrupts, etc.–that are familiar to programmers
- Recognizes specified patterns of behavior through Behavior Specification Units (BSUs) to rapidly categorize behaviors

Potential Applications

- Supports discovery of malicious, incorrect, and unintended behavior in compiled software–no matter how developed
- Supports detect "sleeper" code and vulnerabilities in acquired software
- Enables verification of software functionality
- Provides new artifacts useful for reverse engineering of malwarear

Patent

Kirk D. Sayre and Richard A. Willems, *Behavior Specification and Finding Main,* US Patent application no. 62034410, filed August 7, 2014.

Stephen L. Lindberg, *Dynamic Function Call Graph Visualization,* US Patent application no. 62011125, filed June 12, 2014.

Stacy J. Prowell, Kirk D. Sayre, and Rima L. Awad, *Automated Clustering of Malware Variants Based on Structured Control Flow*, Provisional US Patent Application 62/170,758, filed on June 4, 2015.

Inventor Point of Contact

Stacy Prowell
Computational Sciences and Engineering Division
Oak Ridge National Laboratory

Licensing Contact

David L. Sims
Technology Commercialization Manager
Technology Commercialization
UT-Battelle, LLC
Oak Ridge National Laboratory
Office Phone: 865.241.3808
E-mail: simsdl@ornl.gov

To view this and other ORNL inventions, visit ORNL Partnerships at
http://www.ornl.gov/connect-with-ornl/for-industry/partnerships/technology-licensing/available-technologies

OAK RIDGE National Laboratory
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY
08.2015
ORNL 2015-G00319/tcc