Behavioral Approaches to Software Security

Statement of Research Interests

Mihai Christodorescu

My current research interests are in computer security, with emphasis on software-security assurance. The goal is to be able to answer the questions “Will running this program compromise my system or my data?” and “How does this program differ from programs I trust?”. I believe that a combination of methods from multiple research areas—programming languages, formal methods, and operating systems—will prove fundamental in establishing trust in software. My current work builds on formal program semantics [1, 2], static program analysis [1, 3, 4, 5], decision procedures [1, 5], and dynamic program analysis [6, 7]. Additionally, I have worked on novel topics in software protection [8, 9] and intrusion detection [10, 11].

Current Research

To address the two software-assurance questions above, I have researched and developed a framework for the analysis and detection of malware such as viruses and worms, providing significantly better detection capabilities than state-of-the-art commercial detectors. My dissertation introduces a behavior-based malware detector that determines whether a program contains certain malicious behaviors, thus providing an answer to the first software-assurance question. The answer to the second software-assurance question is a set of minimal behaviors unique to a malicious program, mined out of the malware sample using analysis techniques I have developed.

A malicious behavior is a self-contained, semantically-meaningful interaction with the system. My research takes advantage of the fact that malicious behaviors stay the same across multiple malware variants (because the attackers’ goals rarely change) even though their syntax can vary through obfuscation. My current research answers three fundamental questions.

1. Is it feasible to detect malicious behaviors in an obfuscated binary program? To answer this question, I have developed a formal model for malware detection that defines a language for specifications of malicious behavior (malspecs) with respect to a trusted computing base (TCB). My approach is to distill a program behavior into two related components, a TCB footprint and a set of semantic constraints [1, 3]. The footprint is a set of TCB operations that the malware must invoke in order to achieve its goals, while the semantic constraints restrict the values of arguments to these operations. For example, if the TCB is the operating system, the footprint describes the system calls that appear in the malicious behavior and the constraints capture the dependences between system-call arguments as logic formulas. I designed the malspec language to be expressive and concise. Fewer malspecs are necessary to describe a large set of malware instances, for up to three orders of magnitude reduction from current malware signature schemes. The malspec language also allows formal reasoning about the soundness and completeness of a detector with respect to an obfuscation [2]. By applying this method, I show that my behavior-based malware detection is sound and complete relative to a set of obfuscations commonly used by hackers.

2. Can behavior-based detection be efficient? Based on the formal model of behavior-based detection, I have created a prototype implementation that makes use of decision procedures to determine whether a code fragment is an obfuscated implementation of a malicious behavior. A naïve implementation, using program verification techniques to check for the presence of malicious behavior expressed as a safety property, would be too slow. The observation that different obfuscations generate code patterns with different levels of complexity lead me to develop a chain of decision procedures, from fast and simplistic to slow and powerful ones [1]. The result is a malware-detection prototype that is up to 85% faster than traditional verification techniques, approaching the time range of current commercial detectors that use scan strings. Additional performance improvements are possible and I plan to pursue them as part of my future work. Two possible areas of improvement are the use of incremental analysis techniques to reduce the amount of code inspected during detection, and the combination of static and dynamic techniques to trade off analysis overhead for runtime overhead.

3. How do we extract a specification of malicious behavior from known malware? Manually creating a malspec requires expert knowledge and can be tedious. To alleviate this issue, I have developed an algorithm to derive a specification of malicious behavior from known malware. My technique for mining malspecs identifies behaviors that appear in the malicious program but not in any benign programs, through differential analysis of execution traces. Current results indicate that the malspec-mining algorithm finds 90% of the malspecs described by human experts [7]. The remaining 10% require finer-grained analysis (e.g., information-flow analysis) of the malicious and benign programs. I plan to continue this
line of research in the future by focusing on machine-learning approaches to extract common behavioral features from multiple executions of multiple malware variants.

**Research Philosophy**

My approach to research: (1) addresses practical problems in theoretically sound ways; (2) is interdisciplinary; (3) builds on individual work and collaborative interaction; and (4) involves building effective systems.

**Future Agenda**

Although it is not possible to fully verify the security of complex software, I believe that two distinct approaches can mitigate this problem. First, we need to answer the question “How do we distinguish part-malicious programs from benign programs?” by developing finer-grained detection tools. Second, the question “How do we design large systems to be secure by construction?” can be tackled by creating self-protecting systems that respond intelligently to requests from malicious programs.

**Fine-grained malicious-behavior detection.** A recent trend in the malware world is the rise of part-malicious and part-benign software, such as spyware browser toolbars, that provide some useful functionality to the user while reporting private user data to an external entity. Detecting such malware requires techniques to separate the interactions the program has with the operating systems into independent sets, based on detailed information-flow facts and causal relationships. In the near term, I plan to extend my work on malware detection to finer-grained policies supported by more accurate detection techniques.

1. Can we detect malicious behavior that will happen, rather than malicious behavior that may happen? Detection systems that use runtime monitoring provide precise results, but only after the malicious act happens. Static detectors tell the user ahead of time about the malicious behavior, but they only guarantee that the program might be malicious. The research opportunity here is to design a hybrid detection system with the foresight of a static analyzer and the accuracy of a dynamic analyzer. Such a system could decide whether the current path taken by the program leads to malicious behavior, thus allowing the use of the benign functionality of a program while blocking the malicious part.

2. How (dis)similar are malware families? Malware is by definition behaviorally distinct from benign programs. Yet it is not clear how closely related different malware families are. A system to characterize and classify malware by building on behavioral-distance metrics may lend insight into the evolution of various malware families and indicate new approaches and mechanisms for malware prevention. I envision a technique that combines static and dynamic analyses to create a complete characterization of a malware program in terms of both control-flow and information-flow facts.

**Self-protecting systems through adaptation and collaboration.** My long-term focus is to enhance software systems with new mechanisms for self-protection according to a security policy, which could ban, for example, all self-propagating programs (e.g., viruses). A simplistic enforcement of this policy (e.g., by blocking the program from reading its executable file from disk) can be both too broad as it would block some benign programs and too limited as it would allow some self-propagating programs to succeed. The main research agenda is to find ways to define and then refine such policies and to develop corresponding mechanisms for policy enforcement. Some limited steps in this direction have already been taken, but without integrating the policy into the protected system [12].

1. Can the interface to the protected system be automatically altered to implement a particular security policy? The design and implementation of interfaces to complex systems have been traditionally driven by performance, reliability, and flexibility requirements, with security as a secondary concern. I believe support for security enforcement has to be an integral part of any such system. Similar to how resource usage is adjusted in response to client requests, according to a usage policy (e.g., fairness), access to security-sensitive information resources should be adjusted in response to client requests, according to a security policy. The research problems here are to design the security primitives for expressing adaptive security policies and to implement them efficiently.

2. Can an individual system detect and react to network-wide malicious behavior? In recent years, attackers have proved to be quite adept at using the Internet for distributed malicious computing (e.g., botnets connect many victim computers to one attacker). As a result, it is no longer sufficient to ensure the integrity of individual hosts, as seemingly benign local behaviors can aggregate into network-wide attacks. If we treat the Internet as one (distributed) system, detecting and responding to threats locally, at the host level, are difficult tasks because of the limited network view of an individual host. The results of locally analyzing a program for its impact on the network have to be combined with reasoning about the distributed nature of multi-host behaviors. The research goals in this case range from constructing
the appropriate formalisms for detection of Internet-scale malicious behaviors to identifying the minimal number of hosts that need to collaborate to detect and counter the malicious activity.

References


