On-Time and Scalable Intrusion Detection in Embedded Systems

Albert Mo Kim Cheng
Real-Time Systems Laboratory
Department of Computer Science
University of Houston, TX 77204, USA
cheng@cs.uh.edu

Abstract

Embedded systems are becoming ubiquitous and are increasingly interconnected or networked, making them more vulnerable to security attacks. A large class of these systems such as SCADA and PCS has real-time and safety constraints. Therefore, in addition to satisfying these requirements, achieving system security emerges as a critical challenge to ensure that users can trust these embedded systems to perform correct operations. One objective in a secure system is to identify attacks by detecting anomalous system behaviors. This paper describes the challenges in the design and implementation of such intrusion detection capability.

To deploy a successful intrusion detection system (IDS) especially in an embedded system (the host), we must address (1) accuracy: the IDS identifies no or as few false positives as the resource (time, space, power, etc.) and/or policy constraints allow, and no or as few false negatives as the resource and/or policy constraints allow; (2) efficiency/timeliness: the IDS does not violate the host embedded system's application deadlines and has a reasonable space overhead; (3) scalability: the IDS can scale to work with large embedded systems; and (4) power-awareness: the IDS does not significantly reduce the operational period of battery-powered embedded systems. The paper concludes with an outline of one of several promising embedded IDS approaches under investigation.

1. Introduction

As computer systems are embedded in more devices and their application programs become more complex, they are becoming more vulnerable to attacks which aim to subvert their operations. Many of these systems are wireless devices, making them more susceptible to interference and attacks. It is therefore not acceptable to only satisfy logical correctness and timing constraints in these embedded systems, but also imperative to guarantee a certain level of security so that such systems can be trusted in their proper operations. Of particular interest is to integrate an intrusion detection capability in a PCS (Process Control System) and SCADA (Supervisory Control And Data Acquisition), which is the supervisory control software serving as an interface to the controlled hardware. Also, intrusion detection is required in a sensor network's base station, which is assumed to be the source of all legal messages and hence must not be compromised if protocols for detecting denial-of-message attacks [10] are to work properly.

*Supported in part by the Institute for Space Systems Operations.

This paper focuses on the challenges in implementing host-based [7,14] and network-based [12] intrusion detection systems (IDSs), whose aim is to identify attacks which attempt to subvert processes executing in these embedded systems. The tasks causing the malicious events are then terminated. The remainder of the paper is organized as follows. In section 2, we first describe the problem of specifying legal and malicious behaviors. In section 3, we discuss the design challenges in building embedded IDSs. Section 4 concludes the paper by proposing the rule-based specification and analysis approach, one of several promising approaches under investigation.

2. Specification of Legal Behaviors

One major problem is how to specify/represent the execution behavior of the running processes. In one extreme where the representation is the most accurate, one can simply use the actual code of the running processes. Monitoring is then performed on-the-fly directly on the running processes. Doing so would require prohibitive overheads in terms of time and space, leading to process deadline violations. Also, a large part of this actual-code representation is unnecessary since they are not relevant to the detection of the potential attacks. The other extreme is over-abstraction of the execution behavior that would cause the monitoring procedure to miss many attacks, that is, there would be many undetected positives. Consequently, one must consider these issues in selecting the right representation. Current models [7,14] cannot effectively handle recursion, path sensitivity, and events granularity, so better representations and monitoring need to be developed. Furthermore, different representations may be needed for different systems and/or application domains.

A related problem is malware detection. A malware instance is a program with malicious intent, such as a virus, worm, or trojan [11]. In this problem, the malicious behavior is the result of the execution of a malware instance, and not just a sequence of system calls deviating from a normal behavior. Therefore, it is necessary to specify the signatures or forms of the malware. The question is whether we can adapt or extend the behavioral specification language developed for the IDS to specify malicious behavior of programs that would capture not only the form of a single malware instance, but an entire class of related instances [6]. This would minimize the need for constant updates of the malware detector.

3. Design Challenges

Another key problem is how to monitor the execution behavior once a proper representation is chosen. Overly strict correspondence to the stored legal behavior specification may lead to false positives, and being too loose may lead to false negatives (attacks go undetected). To deploy a successful intrusion detection system, especially in an embedded environment such as SCADA, we must address (1) accuracy, (2) efficiency/timeliness, (3) scalability, and (4) power-awareness.

Accuracy is measured by the number of false negatives and false positives. Ideally, the IDS should not have false negatives and should identify no false positives. That is, no malicious behavior goes undetected, and every malicious behavior detected is a truly malicious behavior. Having no false positives is desirable since these would disrupt normal program executions. However, there are resource constraints including time, memory space, and power. Policy constraints include the importance of the system employing the IDS and the access charge of the IDS.

For example, ensuring no false positives would mean a more detailed specification of all possible legal behaviors, which would incur a prohibitive time, space, and power overhead in checking the observed/monitored behaviors against stored representations of legal behaviors. Thus it would be more practical if the IDS aims for as few false positives as the resource and/or policy constraints allow. Obtaining the proper balance among all these constraints (and quantifying it) suitable for a specific application is a key research topic.

Efficiency is measured by the runtime overhead of monitoring. In embedded/real-time systems, timeliness is critical, that is, the IDS should not violate the host embedded system or network base station's application deadlines and has a reasonable space overhead. How can the schedulability analysis and scheduling [1] be integrated with the scheduling of the host system is an important challenge for the widespread and successful deployment of IDSs in embedded systems.

Scalability refers to the ability of the IDS to work with increasingly large, complex, and networked embedded systems. Here, the challenge is to use specifications and a monitoring procedure whose footprint and runtime overhead grow linearly (or close to this) proportional to the size and complexity of the embedded applications.

Power-awareness concerns with the IDS being able to conserve power in battery-powered embedded systems. It is necessary for a major part of the IDS to remain in a "sleep" mode most of the time and to be awaken responsively when the first sign of a malicious behavior is detected. How to design such an IDS framework is another challenge. How to automatically vary the IDS performance according to a given power budget is an important research problem in embedded/sensor systems.

4. Rule-Based Approach

We have pioneered [3] an ultra-fast semantics-based analysis technique for real-time monitoring and control rule-based systems to determine whether their execution times are bounded and their corresponding worst-case execution times. We also introduced [2] the derivations of behavioral constraint assertions called special forms to characterize bounded behaviors of these systems. We have applied this technique to OPS5 [5] and developed self-stabilizing transformations to OPS5 code [4]. One of several approaches being explored to tackle the above issues is to automatically convert the monitored system behavior into a rule-based program [13], which is then analyzed using the static analysis algorithm in [3] at runtime. We are investigating whether the rule-based specification can model the malicious behaviors that are not effectively modeled by existing techniques. It is especially useful for modeling distributed behaviors [4].

The behavioral constraint assertions encode the legal system behaviors, which are used to determine whether the generated rule-based program deviates from the norm. Since these assertions (special forms) are semantics-based, one special form can cover variations of a legal behavior, making the analysis faster. Furthermore, our recent work on runtime rule-based optimization [8,9] can yield very compact rule-based specifications/programs of the changing system behavior under surveillance.

References

[1] A. M. K. Cheng, Real-time systems: scheduling, analysis and verification, Wiley-Interscience, 2002; 2nd printing with updates, 2005.

[2] A. M. K. Cheng, J. C. Browne, A. K. Mok, and R. H. Wang, "Analysis of Real-Time Rule-Based Systems with Behavioral Constraint Assertions Specified in Estella," IEEE Transactions on Software Engineering, Vol.19, No.9, pp.863-885, Sept. 1993.

[3] A. M. K. Cheng and C.-K. Wang, "Fast Static Analysis of Real-Time Rule-Based Systems to Verify Their Fixed Point Convergence," Proc. 5th IEEE Conf. on Computer Assurance, U.S. National Institute of Standards and Technology, Gaithersburg, MD, pp. 46-56, June 1990.

[4] A. M. K. Cheng and S. Fujii, "Self-Stabilizing Real-Time OPS5 Production Systems," IEEE Transactions on Knowledge & Data Engineering, Vol. 16, No. 12, pp.1543-1554, Dec. 2004.

[5] A. M. K. Cheng and H.-Y. Tsai, "A Graph-Based Approach for Timing Analysis and Refinement of OPS5 Knowledge-Based Systems," IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 2, pages 271-288, February 2004.

[6] M. Christodorescu et al, "Semantics-Aware Malware Detection," Proc. IEEE Symp. on Security & Privacy, Oakland, CA, May 2005.

[7] R. Gopalakrishna, E. H. Spafford, and J. Vitek, "Efficient Intrusion Detection Automaton Inlining," Proc. IEEE Symp. on Security & Privacy, Oakland, CA, May 2005.

[8] J. A. Kang and A. M. K. Cheng, "Shortening Matching Time in OPS5 Production Systems," IEEE Transactions on Software Engineering, Vol. 30, No. 7, pp. 448-457, July 2004.

[9] Y.-H. Lee and A. M. K. Cheng, "Optimizing Real-Time Equational Rule-Based Systems," IEEE Transactions on Software Engineering, Vol. 30, No. 2, pp. 112-125, Feb. 2004.

[10] J. M. McCune, "Detection of Denial-of-Message Attacks on Sensor Network Broadcasts," Proc. IEEE Symp. on Security & Privacy, Oakland, CA, May 2005.

[11] G. McGraw and G. Morrisett, "Attacking malicious code: a report to the infosec research council," IEEE Software, Vol. 17, No. 5, 2000.

[12] S. Rubin, S. Jha and B. P. Miller, "Language-Based Generation and Evaluation of NIDS Signatures," Proc. IEEE Symp. Security & Privacy, Oakland, CA, May 2005.

[13] S. Sodhi and A. M. K. Cheng, "Optimizing Timing Analysis and Verification of Embedded Systems using Rule-Based-Analytic Techniques," Proc. WIP Session of IEEE-CS Real-Time Systems Symposium, Cancun, Mexico, Dec. 2003.

[14] D. Wagner and D. Dean, "Intrusion Detection via Static Analysis," Proc. IEEE Symp. on Security & Privacy, 2001.