Software Security Issues in Embedded Systems

Somesh Jha
Computer Sciences Department,
University of Wisconsin, Madison, WI 53706.

1 Introduction

Embedded systems and networks are becoming increasingly prevalent in critical sectors, such as medical and defense sectors. Therefore, malicious or accidental failures in embedded systems can have dire consequences. Hence, the integrity of embedded software infrastructures, such as configuration and code, is of paramount importance. The autonomous nature of embedded systems also poses new challenges in the context of system integrity. Since embedded systems are reactive, unexpected or malicious environment events or can cause failures, which can have dire consequences in critical sectors. Embedded systems and networks also often have to operate autonomously in a dynamic environment. Therefore, an embedded system has to adapt its behavior to the change in environment or the overall goal. Unauthorized or unverified updates to the infrastructure of an embedded system can also compromise its integrity.

In recent years, there have been significant advances in the area of software security. There have been various techniques developed in the context of software security, such as automated signature generation, vulnerability assessment, and detecting malicious behavior. However, all these techniques are not directly applicable in the context of embedded systems because of following reasons:

- *Dynamic and configurable environment:* Embedded systems are generally deployed in environments that are highly dynamic and configurable. For example, the environment of an embedded system deployed in a battlefield is extremely dynamic.
- *Changing functional requirements:* Functional requirements of an embedded system change over time. Functional requirements of an embedded system deployed in a battleship change with mission of the operation.
- *Interconnected network of components:* Frequently an embedded system is a complex network of components. Therefore, a malicious or accidental fault in a component can lead to a complex cascade of events in the network.
- *Recovery is paramount:* Generally, techniques developed in the realm of software security focus on detection and prevention. Embedded systems are frequently deployed in mission critical applications where consequences of failures can be dire. Therefore, recovery from failures is extremely important in the context of embedded systems.

2 Promising Research Directions

Extending existing techniques in software security to handle the four abovementioned characteristics of embedded systems is an important research direction. I will provide details of two such research directions.

- *Vulnerability assessment and prevention in presence of a dynamic environment:* Existing techniques for vulnerability assessment have been developed for systems (such as servers) whose environments are relatively static. Extending dynamic and static analysis techniques for vulnerability assessment and prevention for systems with dynamic environments is a very interesting research direction. I envision that existing techniques will have to be extended to incorporate specification of the environment. In this context, an interesting research direction would be to generate vulnerability signatures [2, 6] for systems with dynamic environments. I envision the signatures in this case will be parametrized by a specification of the environment, i.e., signatures will only be valid if certain environment conditions are satisfied.

- *Recovery from malicious or accidental faults:* As mentioned before an embedded system is a complex network of components. Therefore, a fault in a component can create a ripple of events throughout the network. This makes recovery for embedded systems extremely challenging. A causality graph for an embedded system is a graph where the nodes are events and edges are the causality between events ($e \rightarrow e'$ means that event e can cause event e'). Techniques for discovering a causality graph of an embedded is essentially for recovering from faults. Essentially the effects of a fault can be determined from examining the causality graph. Techniques for constructing attack graphs [1, 5] and alert correlation [3, 4]

References

[1] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *ACM Conference on Computer and Communications Security*, 2002.

[2] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha. Towards automatic generation of vulnerability based signatures. In *IEEE Symposium on Security and Privacy*, pages 21–24, May 2006.

[3] F. Cuppens and A. Mige. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*, 2002.

[4] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *ACM Conference on Computer and Communications Security*, 2002.

[5] O. Sheyner, J. W. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*, 2002.

[6] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *In the Proceedings of ACM SIGCOMM*, Portland, OR, August 2004.