Chapter 7
Runtime Monitoring of IoT Services to Guarantee Properties

ABSTRACT

Ensuring that a critical information infrastructure remains in a safe and secure state is a mandatory requirement. When IoT services related to such an infrastructure are open to the internet, existing execution monitoring technologies do not work well to protect it; internal malware may compromise and subvert the monitoring mechanism itself, and the safety properties will interact with its security property. In this chapter, an isolation-based solution is proposed to enforce property policies for runtime IoT services. First, the issue of isolation-based service trace observation is addressed by establishing and modelling a virtual channel. Then, the issue of isolation-based policy enforcement is discussed, and the incompleteness and inconsistency of trace knowledge observed in the virtual channel is addressed. Finally, physical systems are introduced into the proposed runtime monitors, and the controllability of IoT services is discussed as an example of the enforcement of service properties. Several experiments are carried out to demonstrate this solution.

INTRODUCTION

In existing work on execution monitoring technologies, runtime monitors are often embedded into protected systems to check and modify each action using property policies, or to actively build a communication channel between

DOI: 10.4018/978-1-5225-7622-8.ch007
themselves and the targets (unsatisfactory application actions are passed to the monitors for checking/correction). Internal malware may compromise and subvert both these applications and the monitoring mechanism itself, if these are within the same environment. In order to achieve isolation-based protection for IoT services, certain technologies based on virtual machines (VMs) can be utilised, which can place runtime monitors outside of the environment of the protected target with isolation. Although the protected services and the protection monitor are isolated and have no active communication channel, existing VM introspection methods and semantic reconstruction methods (Jiang et al., 2007) can be used to reconstruct the VM memory based on low-level “0/1” snapshots, and to search for execution information about service actions within this reconstructed memory. Event flows on virtual network bridges managed by a virtual machine monitor (VMM) in the physical host are intercepted as a complementary means of VM introspection. Communication packets from VMs on industrial buses can also be intercepted using an isolation method. This type of isolation-based event observation provides a basis for the use of runtime monitors in isolated running environments.

Unfortunately, the semantic reconstruction method is passive (non-intrusive) and inefficient, and cannot conveniently capture all service events (executed service actions). The communication interception method is not trustworthy over virtual network bridges, since adversaries may forge messages/events or interleave these events in realistic interconnecting environments. In other words, the events observed using isolation methods may not have been produced by the protected services; some of these can be trusted as moving from/to the protected target, while others are uncertain. Isolation-based event observation methods can be abstracted in the form of a virtual channel that is controlled by adversaries who can forge events in the channel or discard them. In order to build an isolation-based runtime monitor for the virtual channel, each event is assigned a level of certainty indicating its trustworthiness, and the monitor operates based on the trustworthiness of these events.

In order to enforce property policies for a target system, the work of Basin et al. (2013), Ligatti and Reddy (2010), and Dolzhenko et al. (2015) adopted automata-based runtime monitors to check the system’s trace and discarded or edited the actions of the system based on the results of this checking. Although the current work uses the methods and ideas of this study, there are several differences, as follows:

1. **Non-Intrusion:** In existing works, protected systems are recompiled to integrate inline runtime monitors, hooks are placed into the targets, or
APT: A Practical Tunneling Architecture for Routing Scalability
www.igi-global.com/chapter/apt-practical-tunneling-architecture-routing/77499?camid=4v1a