Intrusion Detection and Failure Recovery in Sensor Nodes

Harald Vogt, Matthias Ringwald, Mario Strasser
Institute for Pervasive Computing
ETH Zurich, Switzerland
{vogt,mringwal}@inf.ethz.ch, mast@gmx.net

1 Introduction

Intrusion detection systems (IDS) are important security tools in computer networks. There exist many approaches to this problem in traditional computer networks and wireless ad-hoc networks [ZL00], but literature on this topic with regard to sensor networks is scarce. The goal of failure recovery is to extend the lifetime of a sensor network by restarting or reprogramming failed or misbehaving nodes. In combination, these two measures raise the cost for a potential attacker. Even if an attacker manages to capture a node and abuses it for his own purposes, there is a chance that the aberrant behaviour of this node will be detected and the node be recovered, thus nullifying the attack.

When trying to protect a system from malicious use, it is important to define the goals and capabilities of potential attackers. Here, we consider attackers that try to capture nodes by taking control of the code they are executing. This would allow an attacker to take part in the network's ordinary operation and thus exercise a certain influence on the outcome of its operation, and to exploit the resources of the captured nodes. We will not consider denial-of-service attacks.

There are many possible ways for an attacker to inject malicious code into a node, including the exploitation of weaknesses in its application code or in protocols used for application management, or physical vulnerabilities. The impact of software vulnerabilities can be minimized by using qualitiy assurance tools like code verification and others. Defending against physical attempts at rewriting the application code requires barriers that make access to physical features of the node's hardware as difficult as possible [AK97]. However, all defense mechanisms increase the cost of a sensor network. Therefore, it may be sensible to devote resources to intrustion detection and recovery in order to mitigate the effects of attacks.

Active measures against physical manipulations are also possible. The sensors already built into sensor nodes could help detect physical manipulations. For example, if a node is relocated, acceleration sensors can trigger the zeroization of key material, rendering the node inoperable within the network. In principle, all defense mechanisms can be circumvented, but the required effort should be prohibitively high. Generally, we would like to avoid that attacking a single node becomes cheaper if many nodes have already

been attacked.

In this paper, we sketch a system for detecting intrusions and recovering sensor nodes. We plan to come up with an approach for application-based anomaly specification and detection and node recovery, and a prototypical implementation based on BTnodes [BKR03].

2 Misbehaviour Detection

One of the challenges for an IDS for sensor networks will be to come up with appropriate models for anomaly detection[1]. These models will be governed by the following principles: (1) anomaly detection is based on observations and probing by neighbour nodes; (2) there is no full trust between observer nodes, since they could be under attack themselves; (3) based on the assumed attack patterns (e.g., single nodes, regions), observed data has to be interpreted differently; (4) the specific application of the sensor network determines the modeling of "good" and "bad" behaviour.

Although suitable intrusion models still have to be defined, there is some basic behaviour that can be observed by neighbour nodes which is the basis for any of these models. Canonical conditions on this behaviour can be defined that, if violated, can indicate possible intrusions. Such conditions include: correct forwarding of messages; correspondence between current incoming and outgoing traffic; conformance to communication schedules; integrity of application code. Additionally, one can check for illegal relocation and replication of nodes.

Generally, we rely on nodes to cooperate for detecting misbehaviour. A node that has accumulated enough evidence of a node misbehaving casts a vote in favour of recovery of this node. If a sufficiently large set of nodes participates in the consensus protocol and they agree on recovering the node in question, they start the recovery phase. Emphasis must be put on the efficient implementation of the IDS.

3 Recovery Support

Microcontrollers such as used in the BTnode design include the feature of a bootloader, a piece of software that supports reprogramming "on board", i.e. without the help of external devices. The bootloader accepts application code from a standard communication interface, such as radio or a serial line, and writes it directly to program memory.

To be useful in a malicious environment, the bootloader requires special protection: (1) Before application code is written to memory, its authenticity has to be verified. (2) The keys used for code authentication must reside in a protected memory area that can only be read by the bootloader itself. (3) The bootloader must not be susceptible to manipulation from malicious application code. This can be achieved by placing the bootloader in write-

[1] An alternative approach would be the detection of typical attack signatures. Since there is no empirical data available for sensor networks, we will not consider this approach here.

protected memory. (4) The bootloader must have direct access to the communication interface. This ensures that messages are not intercepted before they reach the bootloader.

These requirements are similar to trusted hardware such as smartcards. The microcontroller employed by the BTnode, Atmel's ATmega128, has built-in support for memory protection, which allows the bootloader to be protected from application code. Rewriting the bootloader code requires erasing all memory, which would effectively delete the keys stored in there. A secure path from the radio interface to the bootloader is currently not directly supported. This issue is addressed by ARM's TrustZone architecture [AF04]. To support all requirements listed above, sensor node designs should be based on such an architecture.

Once a node is being categorized as *failed* by a sufficiently large number of neighbours, they may initiate a recovery procedure by sending authenticated *recovery requests* to the bootloader of the failed node. If the bootloader has collected a certain number of these recovery requests, it switches to a new state in which it is ready to receive application code and rewrite the program memory. After signalling this state change to its neighbours, they start to send the legitimate application code. After a system reset, this code is executed.

This form of recovery is quite expensive compared to ordinary network operation, since all of the application code must be transmitted to the node. The load necessary to send the code could be shared among several neighbours, but the receiving node carries the full load. Also, transmitting the application code can take a significant amount of time if the used interface only provides low bandwidth. Note that no extra space is required for the application code itself, since it can be simply read from the memory of the (correct) nodes.

Although there is usually one main microcontroller unit (MCU) on a sensor node that is responsible for executing application code, there might be some additional MCUs for special tasks, for example handling Bluetooth communications (as the ARM-based Bluetooth module on the BTnode). Some of these MCUs are powerful enough to take on additional tasks like monitoring the main MCU's activities or rewriting the application memory. Since that extra MCU is directly responsible for communication, it is guaranteed that it has access to all data packets as well. Such an extra MCU could also serve as a trusted coprocesser if equipped with special shielding for tamper resistance.

References

[AF04] Tiago Alves and Don Felton. TrustZone: Integrated Hardware and Software Security. Technical report, ARM, July 2004.

[AK97] Ross Anderson and Markus Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *5th Int. Workshop on Security Protocols*, number 1361 in LNCS. Springer-Verlag, 1997.

[BKR03] Jan Beutel, Oliver Kasten, and Matthias Ringwald. BTnodes - A Distributed Platform for Sensor Nodes. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*. ACM Press, 2003.

[ZL00] Yongguang Zhang and Wenke Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In *MOBICOM 2000*. ACM Press, 2000.