Fuzzing Wi-Fi Drivers to Locate Security Vulnerabilities*

Manuel Mendonça, Nuno Ferreira Neves
University of Lisboa, Portugal
manuelmendonca@msn.com, nuno@di.fc.ul.pt

Abstract

Wireless LANs (WLAN) are becoming ubiquitous, as more and more consumer electronic equipments start to support them. This creates new security concerns, since hackers no longer need physical connection to the networks linking the devices, but only need to be in their proximity, to send malicious data to exploit some vulnerability. In this paper we present a fuzzer, called Wdev-Fuzzer, which can be utilized to locate security vulnerabilities in Wi-Fi device drivers. Our preliminary experiments with a Windows Mobile 5 device driver indicate that Wdev-Fuzzer can be quite effective at finding previously unknown problems.

1. Introduction

Wireless LANs give individuals the freedom to stay connected, while moving from one coverage area to another. They can be used to extend a wired infrastructure or to replace existing ones, saving costs not only due to the declining prices of the wireless components, but also because they require no (or simpler) data cable installations. Nowadays WLAN technologies are becoming ubiquitous, and most consumer electronic products (such as laptops, PADs, cellular phones, video game consoles, digital cameras, printers and video projectors) are equipped with them.

WLAN however introduce newer problems, since they weaken the security perimeter. In many places, like airports and shopping malls, there are dozens of rogue networks just waiting to entrap unsuspecting travelers. Additionally, it is much easier to compromise WLAN equipments because attackers only need to be in proximity of the devices to perform the attack (physical connection to the network is no longer necessary).

Although some security failures are due to wrong system configurations, many result from the exploitation of implementation bugs in the software components. In this paper, we are particularly interested in locating this sort of bugs (or vulnerabilities) in device drivers (DD) of WLAN, to allow their removal. These DD are the entry point of any device, and therefore they are the first software to process the potentially malicious traffic coming from an attacker. Moreover, almost any vulnerability in these DD has a catastrophic impact since they run in the operating system (OS) kernel.

In general, DD provide an abstraction layer between the physical details of equipments and the kernel. Nowadays they are becoming the most dynamic and largest part of an OS. Their design involves knowledge from several disparate areas, like OS internals, chipset details, and synchronization that are not simultaneously mastered by programmers or designers. Therefore, they are hard to implement and to maintain, and many times they end up being deployed with bugs.

The methods for discovering vulnerabilities in DD depend on the availability of the driver code. If the code is public, then source code auditing may lead to good results, as one can read and check for implementation flaws. However, in the majority of situations, the code is closed. In this case, black box testing may be performed, where the functional behavior of the unit under test (UUT) is verified (output results) against the input values that are provided. Reverse engineering may also be employed to discover vulnerabilities, but it is costly, time consuming and demands profound knowledge on system architecture and machine code.

Vulnerabilities can also be discovered by another black box testing methodology, some times called fuzzing [1]. Fuzzing consists on presenting malformed data to the interface of the software component and on observing the outcomes. This technique may require further refinements to catch more complex bugs, due to protocol specificities, but it can be very effective in locating several kinds of vulnerabilities.

In our work, we have designed a new fuzzer architecture that can be used to automatically locate vulnerabilities in WLAN device drivers. The current implementation of the architecture, called Wdev-Fuzzer, supports the Wi-Fi protocol and is specialized

* This work was partially supported by the EU through project IST-4-027513-STP (CRUTIAL) and NoE IST-4- 026764-NOE (RESIST), and by the FCT through project POSC/EIA/61643/2004 (AJECT) and the Large- Scale Informatic Systems Laboratory (LASIGE).

to test PDA equipments. In the future, we intend to extend it to other communication protocols, such as Ird and Bluetooth.

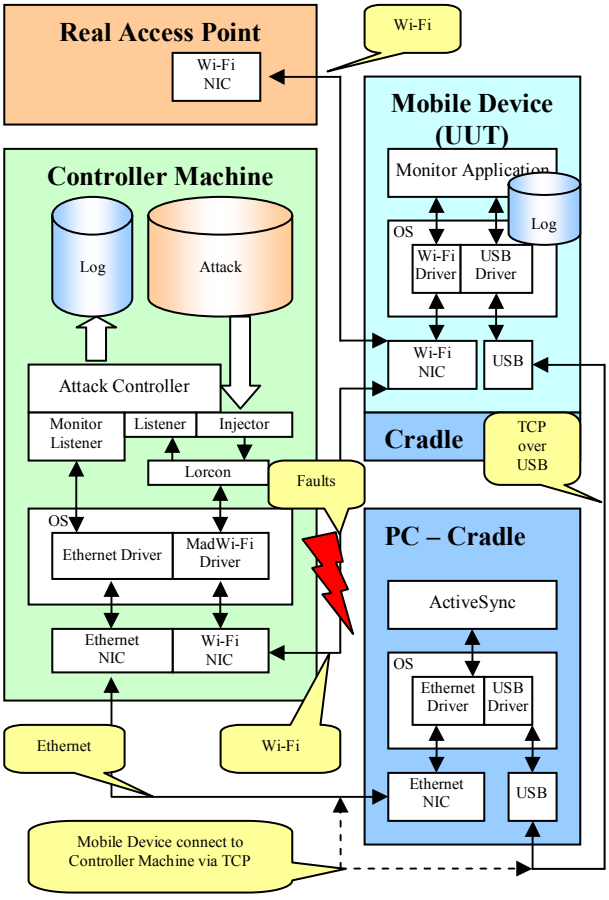

Figure 1. The process of fuzzing Wi-Fi frames.

2. The Wdev-Fuzzer

The Wdev-Fuzzer utilizes the test infrastructure represented in Figure 1, which is composed by 4 elements: a Controller Machine, a Mobile Device (or the UUT), a PC-Cradle, and a Real Access Point. The Controller Machine is responsible for generating Wi-Fi packets containing malicious data (e.g., out-of-bound values, repeated tags) and to send them to the UUT. This element also monitors the outcomes of the tests, and saves the collected data in the disk for future analysis. One of the problems in developing the Controller is to find a card and a driver that allows the injection of raw Wi-Fi frames. Currently, we are using the Linux open source MadWi-Fi [2] driver, together with the Lorcon [3] generic library for injecting the frames.

The UUT is the target Wi-Fi device of the experiments. It runs a Monitor Application that regularly connects to the Monitor Listener of the Controller, informing the current list of detected access points (AP) and the status of any existing connection. This data is especially useful when testing Beacon and Probe frames, as the detection of the AP is crucial to determine the correction of the error handling mechanisms. The Monitor Listener forwards some of the data to the Attack Controller to synchronize the next attack. The Wi-Fi responses transmitted by the UUT are processed by the Packet Listener of the Controller. Each incoming packet is carefully examined to determine any unexpected behavior.

The UUT is physically attached to a Cradle, which in turn is connected to a PC through an USB port. This way, the Monitor Application can reach the Controller through an alternate link, leaving the Wi-Fi medium completely free for the experiments.

To keep the complexity of the code of the Controller manageable, a Real AP is utilized to take the UUT through the various states of the Wi-Fi protocol. This way, specific frames can be injected in every state.

3. Experimental Results

At this moment, we are using Wdev-Fuzzer to locate bugs in the Wi-Fi driver of Windows Mobile 5. The UUT is a HP iPAQ hw6915 PDA. Our preliminary results have already uncovered a few mismatches between the DD implementation and the Wi-Fi standard. For instance, we have noticed that the "Duration" field of the Beacon frames is not being tested properly, since the UUT reports a valid AP even if the field value is above the maximum acceptable value. Additionally, we have found a specific Beacon frame that causes the OS hang of the UUT. We are currently investigating this bug, to determine if it can be remotely exploited.

4. References

[1] P. Oehlert, *Violating Assumptions with Fuzzing*, IEEE Security & Privacy, pages 58-62, March/April 2005.
[2] Atheros, *MadWifi-Multiband Atheros Driver for Wireless Fidelity*, July 2007. http://madwifi.org/
[3] Lorcon Team, *LORCON - Loss Of Radio CONnectivity*, July 2007. http://802.11ninja.net/lorcon