IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 3, Issue 5, Oct -Nov, 2015
**ISSN: 2320 – 8791 (Impact Factor: 2.317)**
**www.ijreat.org**

# A Fuzz Testing Framework for Wi-Fi Devices

**Poonam Joshi[1], Parth Patel[2], Raveena Parikh[3], Riddhi Padte[4], Vishal Padma[5]**

[1,2,3,4,5]**Information technology, Mumbai University, Atharva college of Engineering, Mumbai, Maharashtra 400095, India**

**Abstract**
Wireless LAN has become an integral part of our lives. Almost every organization or institution use WLAN these days. Although WLAN is very convenient to use for different purposes, it has created many threats as an attacker will not have to connect to the wireless LAN through a physical device to directly launch an attack on the target device if he is present in the same wireless LAN. Thus, driver security is critical for Wi-Fi devices and to test the vulnerabilities of a Wi-Fi device, a Fuzzing testing framework has been proposed which will be used for an Android device's Wi-Fi driver. We will basically simulate an attack against the Wi-Fi device to be tested when it will be searching for an access point or already connected to a WLAN and we can determine different cases in which our device is able to handle the malicious packets correctly.

***Keywords:*** *Android, Fuzzing, WLAN, Security, Packet Injection.*

## 1. Introduction

With the field of mobile internet and wireless devices evolving constantly, almost every day we access a wireless network using a laptop or smart phones. WLAN (Wireless Local Area Network) has become an important part of our lives. However, due to the weakening of the WLAN security boundary, there may exist, a hacker carefully camouflaged in a public Wi-Fi network. While connecting our Wi-Fi device to such a corrupted network, we don't come to know whether the network is safe or not and we connect our device to that network eventually allowing the attacker to collect our private information. Some security failures are due to wrong system configurations; many result due to the exploitation of implementation bugs in the software components. The main aim of this paper is to locate these kinds of bugs (or vulnerabilities) in device drivers (DD) of WLAN, and allow their removal. These device drivers are the entry point of any device, and therefore they are the first piece of software to process the potentially malicious traffic coming from an attacker. Moreover, any vulnerability in these DDs can have a drastic impact since they run in the operating system kernel. Fuzzing is a black box testing method to discover vulnerabilities; it has been mainly used for protocol testing, when source code is not available. It has proved to be a very effective method to discover vulnerabilities. This paper aims at designing a Fuzzing framework, with this it will be possible to construct malformed packets, which will be targeted to the wi-fi device, and eventually discover known and unknown vulnerabilities.

## 2. Review of Literature

### 2.1 Fuzzing Wi-Fi Drivers to locate Security vulnerabilities.

This work was carried out in 2007 by Manuel Mendonça, Nuno Ferreira Neves in which they have developed a new fuzzer architecture called Wdev-Fuzzer which is used to locate security vulnerabilities in Wi-Fi device drivers. In this paper, they have given the diagram of the entire process of fuzzing Wi-Fi frames. They have used the Linux open source Madwifi driver with Lorcon to inject frames in the mobile device to be tested. The Wi-Fi responses transmitted by the mobile device are processed by the packet listener and each packet is examined carefully to check for any unexpected behavior. Experimental results have shown that Wdev-Fuzzer can be quite effective in detecting unknown problems.

### 2.5 Stateful Fuzzing of Wireless Device Drivers in an Emulated Environment.

This work was carried out by Sylvester Keil and Clemens Kolbitsch in which they have documented the process of identifying vulnerabilities in IEEE 802.11 device drivers through fuzzing. They have presented a new approach to fuzzing 802.11 device drivers in an emulated environment due to the complexity of 802.11 protocol. In this paper, first the process of creating a virtual 802.11 device for process emulator QEMU is described and then development of a stateful 802.11 fuzzer based on the virtual device is discussed. The goal of this paper was to find a solution for the timing problems encountered in fuzzing the stateful frame types. By using a simulated environment to test the target system and replacing the problematic wireless communication with high-level means of IPC, not only the timing problems were solved but also a number of practical advantages were achieved.

### 2.2 Packet Crafting Using Scapy.

This work is a thesis and was done by William Zereneh. In this paper, the whole process of packet crafting is explained. Packet Crafting consists of four aspects- packet assembly, packet editing, packet replay and packet decoding. The tool used for packet crafting is Scapy. Scapy is an open source network programming language which is based on Python. Scapy is powerful and primitive in such a way that it will allow the crafter to pull the packets off the wire or create ones as required. Once the packets are assembled, the crafter can change the fields of the packets (header or payload) according to his own desire. Such a packet is then played or replayed on to the network as many times and at any speed required as stipulated by the testing case at hand; Scapy can replay such packets. The packets launched on to the network will cause a response to be generated from the target device which will be captured to analyze and understand the problem at hand or to confirm results; Scapy can decode the packets.

## 3. Proposed System

In this new project we propose a new framework that will test the reliability and vulnerability of Wi-Fi device drivers. We will be using a fuzzy logic to inject malicious packets into the driver and check if the device is able to identify it or no. 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.

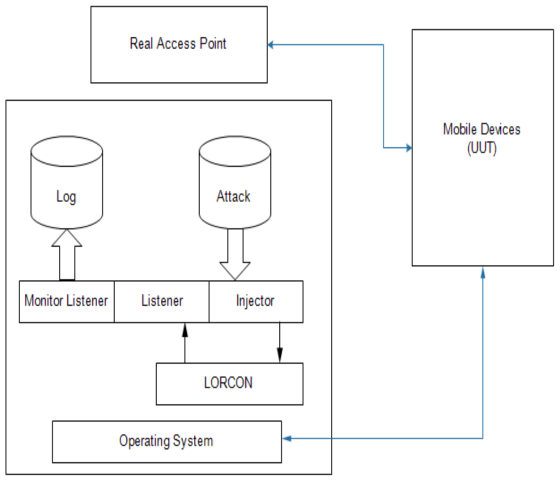

Fig.1 Proposed system.

## 4. Methodology

Our System consists of 5 modules:

### 4.1 Association

It is the first step in connecting to the vulnerable devices which are connected to the internet using Wi-Fi. Association means agree to specific terms in order to connect to a wireless LAN (Wi-Fi). Once association is done the attacker is half succeeded in his task of attacking vulnerable devices connected to the Wi-Fi. For association the attacker will send a probe request (Beacon packet) which contains shell code plus the exploit code in it.

### 4.2 Authentication

It is the second step in which when done completes the connection of the attacker to the vulnerable devices. Authentication is done to check whether the true user is accessing the required information. Once the authentication is done the user or the attacker is connected to the vulnerable or all the devices connected through the wireless LAN (Wi-Fi). After authentication the user or the attacker is connected to the device or devices and they both can start exchanging packets through wireless LAN (Wi-Fi).

### 4.3 De-authentication

De-authentication is basically the opposite of authentication. In authentication the legitimate users are being checked for their identity whether they are the true users or not. In the same way in de-authentication all the illegitimate users are checked for their identity. Here all the users with fake ID's are traced so that they cannot authenticate to any launch an attack and steal the valuable information. De-authentication is very important as all the attackers may try to launch an attack and get the information.

### 4.4 Disassociation

Disassociation is the opposite method of association. In association, normally users agree to some specific terms and then connect to the users not totally but initially. The opposite happens in disassociation where the users are not able to connect or doesn't agree to the terms specified to connect. Here also the legitimate users can be filtered as all the illegitimate users will not agree to the specified terms to connect. Only those users are connected who will agree to the terms or the disassociation occurs where the users doesn't agree to the terms specified.

### 4.5 Packet injection

Packet injection is used in our system to send malicious packets to the targeted device connected to WLAN. Here the random packets will be generated for fuzz testing. If the targeted device accepts the packets then the packet injection will be successful and this will reveal that the targeted device has some vulnerabilities. The tool that will be used to generate the random packets is Scapy.

## 5. Overview of the system

The monitoring system will be connected to the Android mobile device (Unit under test) through WLAN. After the association the beacon packet will be sent. We will be simulating an attack that will send malicious packets to the targeted device. The attack will be simulated using a packet injecting tool and also a listener that will monitor the bugs and vulnerabilities in the targeted device. The library for packet crafting would be LORCON. The bugs detected can be then used to determine the overall security issues of the targeted device.

## 4. Conclusion

Thus, in this paper, we propose to implement a fuzzing framework to test the security of a WI-FI device. This framework will also be able to discover known threats and unknown vulnerabilities of the device. This framework would be largely helpful in industrial, domestic sectors of device testing and military applications. This framework can be further extended to other protocols such as Bluetooth and NFC in future.

## References

[1] Michael Sutton, Fuzzing Brute Force Vulnerability Discovery. Addison-Wesley Professional 2007 25-27.
[2] Gray Hat Python: Python Programming for Hackers and Reverse Engineers, Electronic Industry Press 2011-3,68-73.
[3] Yang Zhanggang, depth analysis of the Android system [M]. Electronic Industry Press .2013, 355-367.
[4] Madwifi driver, June 2007, www.madwifi.org.
[5] Lorcon project, June 2007 http:// 802.11ninja.net/lorcon.
[6] A. Albinet, J.Arlat, and J.C. Fabre, Characterization of the Impact of Faulty Drivers on the Robustness of the Linux Kernel, Proceedings of The International Conference on Dependable Systems and Networks [J] IEEE, 2004.
[7] J. Duraes and H. Madeira, Characterization of Operating System Behavior in the Presence of Faulty Drivers through Software Fault Emulation [J], Proceedings of the Pacific Rim International Symposium on Dependable Computing, pp. 201-209, December 2002.
[8] Manuel Mendonça, Nuno Ferreira Neves, Fuzzing Wi-Fi Drivers to Locate Security Vulnerabilities, CONFERENCE PAPER · DECEMBER 2007 DOI: 10.1109/HASE.2007.43 · Source: IEEE Xplore.
[9] P. Oehlert, Violating Assumptions with Fuzzing, IEEE Security & Privacy, pages 58-62, March/April 2005.
[10] Atheros, MadWifi-Multiband Atheros Driver for Wireless Fidelity, July 2007. http://madwifi.org/.
[11] Lorcon Team, LORCON - Loss Of Radio CONnectivity, July 2007. http://802.11ninja.net/lorcon.
[12] William Zereneh, Packet Crafting Using Scapy, a thesis presented to Ryerson University, Toronto, Ontario, Canada.
[13] Sylvester Keil and Clemens Kolbitsch, Stateful Fuzzing of Wireless Device Drivers in an Emulated Environment, Secure Systems Lab, Technical University Vienna in cooperation with SEC consult.