ACSAC artifact submission abstract

September 20, 2019

1 Abstract

We have provided the source code which was used for the evaluation section of our paper. Specifically, we have submitted all the source code we used to answer the research questions in the paper (RQ1-RQ6). The code includes a JSON file containing a set of grammars for more than 80 AT commands and implementations of our fuzzing framework running over both USB and Bluetooth. To demonstrate the effectiveness of our proposed mutation, crossover, and feedback mechanisms, we have provided two versions of ATFuzzer for each of these mechanisms. For instance, we will provide one version with mutation and another one without mutation to compare the effectiveness of our proposed mutation techniques. We have also provided our version of AFL with five mutation strategies (walking bit flips, walking byte flips, known Integers, block deletion, and block swapping). In order to conduct the experiments thoroughly, it is necessary to acquire a modest number of devices (refer to Table: 3 of the paper). Moreover, for running ATFuzzer over USB, most of the devices have to be specifically configured. We will provide a documentation to configure them correctly. For running ATFuzzer over Bluetooth, either a Bluetooth enabled laptop, or a Bluetooth dongle is necessary. All the source code is developed using **Python 2.7.15**, and it is designed to run on Windows and Linux operating systems only.

2 Github Repository

https://github.com/Imtiazkarimik23/ATFuzzer

2.1 Structure of ATFuzzer Implementation

In the following we provide a description of the structure of the implementation of ATFuzzer.

commandGrammar.json. The JSON file contains a set of grammars for more than 80 AT commands. The grammars are defined following a specific structure that allows the program to translate them efficiently. Specifically, the structure is constituted of the command name and the list of allowed parameters. Hence, for each parameter the accepted values are given. For instance, the structure of the AT command `AT+CFUN` is the following:

```
structure: "cmd", "equal", "fun", "rst",
cmd: "+CFUN",
equal: "="
fun: integer in range [0, 127]
rst: integer in range [0, 1]
```

.

executeFuzzer.py. Main program which allows the user to run ATFuzzer. It provides different options of execution (as described in 3). The program takes 3 input arguments:

- List of AT commands to test, or the word "multi" to let the program randomly chose the AT commands;
- The name of the target device (mostly used to identify the associated result file);
- Computer port employed in the communication with the AT interface of the target device (e.g., `ttyACM0` in Linux, `COM3` in Windows). This argument is optional, that is, if the argument is omitted, the program can automatically detect the correct port.

grammarFuzzer.py. It reads the grammars for the AT commands submitted by the user and performs the fuzzing loop which consists of 4 main steps:

- *input generation*: given the input grammars, it generates a set of 10 AT command instances;
- *input submission*: submits the generated input to the AT interface of the target device through the functions implemented in `atCmdInterface.py`;
- *grammar evaluation*: based on the result of the input submission, it evaluates and selects the grammars for the next round;
- *grammar evolution*: performs grammar crossover and grammar mutation operation through the functions implemented in `grammarModifier.py`.

multiGrammarFuzzer.py. Implements the actual fuzzing. It reads the grammars for the randomly chosen AT commands and performs the fuzzing loop: input generation - input submission - grammar evaluation - grammar evolution. The only difference with `grammarFuzzer.py` is that set of input AT commands is randomly generated by the program, instead of being manually inserted by the user.

grammarModifier.py. Implements the functions for the evolution phase. Such functions include *grammar crossover* and *grammar mutation*. The former swaps the fields of the production rules among the given parent grammars. The latter mutates a given grammar G by randomly inserting/adding a field chosen from the production rule of G at a random location, trimming an argument from a production rule for G and negating the conditions associated to a randomly picked production rule for G.

inputGen.py. Generates a random AT command instance given an input grammar.

atCmdInterface.py. Implements the functions necessary to interact with the AT interface of the target device. It is responsible for setting up the communication with the device, based on the fuzzing channel initially selected (USB or Bluetooth), submitting the AT command instances, and finally collecting and evaluating the responses. If anyone wants to test a specific command instead of fuzzing, it is possible to do so using atCmdInterface.py. In the main function of this file the user can call usb_fuzz or $bluetooth\_fuzz$ with the specific AT command he or she is willing to test. In that case the user will have to provide the device name and Bluetooth address as arguments.

afl_fuzzer.py. Implements the functions used to execute AFL fuzzer in the context of AT commands. Specifically, it mutates the input with five standard AFL operations: `bit_flip(cmd)`, `byte_flip(cmd)`, `known_integer(cmd`, `block_deletion(cmd)`, `block_swapping(cmd)`.

utilityFunctions.py. Implements support functions for the execution of the main program, such as customized random method, functions for exploring given grammars and so on.

results. Directory containing the output of each ATFuzzer execution. The name of the device given as input of `executeFuzzer.py` is used as name of the file with the related results.

3 How to Run

3.1 Requirements

Python 2.7.15. Please do not use python 3 because there are library incompatibilities. The required libraries are specified in the file `requirements.txt` and they can be installed executing the command `pip install -r requirements.txt`.

Note: the latest version of the module `pybluez` for python 2.7 is not compatible with Windows. To install `pybluez` on Windows it is necessary to download a previous version. Download `PyBluez0.22cp27` at:

```
www.lfd.uci.edu/~gohlke/pythonlibs/#pybluez
```

and install the module with the command:

```
pip install <pybluez file.whl>.
```

3.2 Run the Code

To run ATFuzzer execute the following command:

`sudo ./executeFuzzer.py <list of grammars> <device_name> <port (optional)>`.

Alternatively, it is possible to execute ATFuzzer with multiple random chosen grammars with the command:

`sudo ./executeFuzzer.py multi <device_name> <port (optional)>`.

The program then asks to choose among 4 option:

- 0 - Standard fuzzer (includes crossover, mutation, feedback evaluation)
- 1 - No feedback fuzzer
- 2 - No crossover fuzzer
- 3 - No mutation fuzzer

These options allows the user to choose which type of ATFuzzer to run. This is fundamental to test and evaluate the effectiveness of our fuzzer.

Finally, ATFuzzer requires to specify which channel will be used for the AT commands transmission. It is possible to select one among three options:

- b - Bluetooth
- u - USB
- t - Test execution (does not require any connected device)

If the `Bluetooth` option is selected, the program asks for the Bluetooth address of the target device. The user may insert the Bluetooth MAC address of the device in the specific format: `XX:XX:XX:XX:XX:XX` (e.g., `1A:2B:3C:4D:5E:6F`).

`Test execution` executes ATFuzzer with fake evaluation parameters and without submitting any command to a device. This option is only for testing purpose, so do not use it to fuzz an actual smartphone.

Note: if you run Bluetooth ATFuzzer on Linux, it may be necessary to execute the program with `sudo`, depending on the system configuration and if you run it on Windows you need to run cmd as Administrator.

4 Running AT Command over Bluetooth:

Running AT commands over Bluetooth does not require any configuration on the phone. The only requirement is to pair the phone and a Bluetooth enabled laptop or Bluetooth dongle is needed. However, the phone has to be connected through USB to the device running the code to detect disruptions in cellular network and internet connectivity.

5 Running AT commands over USB:

Unlike Bluetooth, running AT commands over USB requires certain configurations on the phone. We tested it on 8 phones and will provide the smartphone configuration instructions in as detail as possible. Moreover, if applied to a different phone, ATFuzzer can behave either two ways: (i) It works properly (ii) It reboots constantly (repeatedly starts the daemon). The second case can mean one of the two things, either the device is *not configured properly* or it is *not possible to run* AT commands over USB for that specific device.

6 Smartphone Configuration Instructions

We provide the fundamental steps to set up some of the smartphones we use in our experiment. One crucial requirement is to have `adb` installed on the computer! (Windows also requires to install the `adb` drivers)

6.1 Nexus 6 and Nexus 6P

For Nexus6 and Nexus 6P the following steps are required:

1. Unlock Developer options in phone settings and enable USB debugging.
2. Connect the phone to the computer and run the following commands (it can be done both on Linux and Windows):
   - `adb reboot fastboot`
   - `fastboot oem config bootmode bp-tools (Nexus 6)`
   - `fastboot oem enable-bp-tools (Nexus 6P)`
   - `fastboot reboot`

   This sets the property sys.usb.config to diag,serial_smd,rmnet_ipa,adb and allows the user to send AT commands through the AT interface.

Windows might requires to install/reinstall some of the drivers. Check the environment with the following steps:

1. On Control Panel > Hardware and Sound > Device and Printers, double click Nexus 6 (Nexus 6P) icon. The properties window should open.

2. On the Hardware panel of the property window the following devices should be listed:

   - Android Composite ADB Interface
   - Qualcomm HS-USB Diagnostics
   - Qualcomm HS-USB Modem
   - Qualcomm Wireless HS-USB Ethernet Adapter
   - USB Composite Devide

   Qualcomm Modem device is the one needed to use AT Commands. On the device's properties Modem panel, the used Port (e.g. COM3) and the Maximum Port Speed are displayed.

   If Qualcomm Modem is not listed and only Nexus 6 (Nexus 6P) is listed, one problem could be Windows drivers. In this case it is necessary to unistall the drivers for all the Nexus 6 (Nexus 6P) devices listed (more then one could be listed). Once the previous step is completed, reconnect the phone to the computer and wait for the drivers to be automatically installed.

6.2 Nexus 5

To configure the phone in order to use AT Commands the following steps are required:

1. Root the phone.
2. Connect the phone to the computer and go to the super user interface through adb and run the command:

   - `setprop sys.usb.config diag,adb`

   This will set the property sys.usb.config to diag,adb and allow the user to send AT commands through the AT interface.

6.3 Samsung Galaxy S8 Plus, Note 2, S3 and LG G3

For Samsung Galaxy s8 Plus, Note 2, S3 and LG G3 plus it is possible to switch the USB interface to inject AT command through usbswitcher. After running usbswitcher AT command can be connected through establishing a serial connection to */dev/ttyACM**. Few things to be noted here for Note 2 and S3 there is a command whitelisting implemented therefore for most of the time the fuzzer will respond either "OK" or "ERROR", which is not the typical behaviour.

7 Execution Time Estimate

For the smartphones listed on the paper we have run our fuzzer over USB and Bluetooth each for one month. To answer RQ3-RQ6 we have run our version of AFL with ATFuzzer w/o feedback, ATFuzzer w/o crossover and ATFuzzer w/o mutation each for 3 days with Nexus 5.

8 Running Example:

8.1 Bluetooth:

This example is run on linux with nexus6.

```
    cyber2slab@cyber2slab-ThinkPad-T480:~/ATFuzzer$ sudo ./executeFuzzer.py multi
    nexus6

 --- Select which type of fuzzer you want to execute ---
 > 0 - Standard fuzzer (includes crossover, mutation, feedback evaluation)
 > 1 - No feedback fuzzer
 > 2 - No crossover fuzzer
 > 3 - No mutation fuzzer
0

 --- Select which channel you want to use for fuzzing ---
 > b - Bluetooth
 > u - USB
 > t - Test execution (does not require any connected device)
b

 You have selected Bluetooth fuzzer.
 Please insert the Bluetooth MAC address of the target device.
 (Use the format XX:XX:XX:XX:XX:XX)
e4:90:7e:ee:2b:84

 --- Executing ATFuzzer! ---

('Fuzzing grammars: ', "[u'+CREG', u'D', u'+CPNET']")
AT+CREG=0;D89898046311;;+CPNET=1
2019-09-19 15:28:51.969112

ERROR

Input: AT+CREG=0;D89898046311;;+CPNET=1 Output:
ERROR
flag is now 0
```

```
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
flag is now 0
[0.4514935314655304, 0]
AT+CREG=0;D6956496992;;+CPNET=2
2019-09-19 15:29:34.749287

ERROR
```

The fuzzer will run this way up to the time the user wants to run it, and if for any grammar there is any disruption the flag will be set to 1. In that case, the grammar along with the input will be saved in the "results" folder.

9 USB:

This is an example of running Nexus 6 over USB.

```
    C:\Users\purdu\Downloads\ATFuzzer-master>executeFuzzer.py multi nexus6

 --- Select which type of fuzzer you want to execute ---
 > 0 - Standard fuzzer (includes crossover, mutation, feedback evaluation)
 > 1 - No feedback fuzzer
 > 2 - No crossover fuzzer
 > 3 - No mutation fuzzer
0

 --- Select which channel you want to use for fuzzing ---
 > b - Bluetooth
 > u - USB
 > t - Test execution (does not require any connected device)
u

 --- Executing ATFuzzer! ---

('Fuzzing grammars: ', "[u'+DEVCONINFO', u'+COLP', u'+CRLP']")
Probing for COM ports...
Trying device  COM22
0.00400018692017
Serial port:  COM22
+DEVCONINFO/9"D7<;+CLIR=4:W,}i;+CRLP=64991191,71588568,5715157,98191,1880627364,
D7/u-bBp2019-09-19 18:08:47.748000
```