A Fuzz system of the SMS module on cellphones

Fu Zidan, Zhang Quan

School of Electronic Science and Engineering, National University of Defense Technology, Changsha, 410073, China

email: 513261490@qq.com

Keywords: vulnerability testing; SMS; Fuzz;

Abstract. Since the mobile network become popular, threats and challenges upon the security of cellphone are more serious than ever. However, the vulnerability testing methods are limited in the old white box testing which the manufacturers adopt basing on the knowledge from the original information of designing the cellphone. There is a new cooperation testing system which is different from the traditional ones and can run black box testing on any cellphone's short message module. This may inspire more testing engineers to be engaged in the vulnerabilities testing of the cellphone.

Introduction

The birth of 3G, 4G and smart phone changed the world. The working style of the society is transforming from the traditional desktop working and network on computer to cloud computing and mobile office, which accelerates the speed and efficiency of the society.[1] Because of the convenience and applicability, the mobile technology tend to be much more popular, ranging from all kinds of products. At the same time, plenty of new ways to attack the mobile device and local wireless network are found by hackers. The fast development of the mobile network and the stressing of the economical benefit made the company focus on speed rather than quality, especially the work on the safety of the products. Today, the threats and challenges in mobile network is no less than that of the traditional Internet.

Most original testing method of the cellphone, especially some safety testing and pressure testing aiming at the firmware, are done by the manufacturers. They own all the related information and documents of their products, and as the designers, they are very familiar with the places where problems always occur. Therefore, this kind of white box testing can be easily done. However, the shortcomings of this kind of testing are obvious too. First, testing engineers are limited in the few designers of the phone, which makes the tests slow, narrow and incomplete. This kind of tests can't cover all the vulnerabilities. And sometimes the docs with the transcendental knowledge of the phone can misguide the engineers and make them ignore some places where real vulnerabilities can hide. Second, because of the differences among the phones, a testing method in company A probably can't be effective with a cellphone produced by company B. Third, this kind of testing focus more of the functional deficiency rather than safety deficiency.

This article includes a method which can be used to launch fuzz testing for every device which contains a SMS module.

Analysis of 3GPP message format

According to the standard 3GPP communication protocol, messages are transmitted by 2 ways of coding in the mobile network:

a. Protocol Data Unit

b. Text Mode

In text mode, the data is simply "as the message is", while PDU mode need the original payload be coded in a complicated way and be transmitted through the IMS（IP Multimedia Subsystem）. The coding process made the message more standard and functional. New added functions include some simple voice, animation, and all these things need proper configuration of controlling words

in the coded message.[2][3] These advantages made the PDU mode the most popular way to transmit messages. At the same time, various functions and complicated configurations make it vulnerable too. It is very possible that a mistake of a single bit can turn the message an unrecognized rubbish, even an evil one which can crash the software. Considering the popularity of the two mode, this article will aim at the messages transmitted in PDU mode.

From the official docs, we can acquire the specific structure of the message and split it into several segments. After that, we can give an assessment of each segments and find out how the bad words can ruin the system, which is the basic factors to help us generate the testing samples. And then, with the help of our devices, we can control every single bit in each message for test.

Now we will give a brief introduction of the segments in a standard PDU-coded message:

Chart.1 Architecture of RPDU message

RP-Message Type	3 bits
RP-Message Reference	1 octet
RP-Originator Address	1-12 octets
RP-Destination Address	1 octet
RP-User Data	<=233 octets

Chart 1 shows the overall structure of a PDU-coded message. RP-MT stands for the type of the message, RP-MR for some related information of controlling messages between the device and the base station, RP-OA for the phone number information of sender，RP-DA for phone number information of receiver，and RP-UD for the real payload of the message.

Chart.2 Specific architecture of the RP-UP part

UDL	1 octet
OA	1-12 octets
PID	1 octet
DCS	1 octet
TIME INFO	7 octet
PAYLOAD_LENGTH	1 octet
PAYLOAD	<=140 octets

Chart 2 shows the specific structure of RP-UD.PID and DCS represent protocol identity and data coding style respectively. TP-VP represents the length of valid period from the base station's receiving to the mobile device's receiving. TIME INFO represents the 7 octets' information about when the message is sent by the base station. TP-UD represents the real text content of the message, while TP-UDL represents the length of the TP-UD. The size of TP-UD can't exceed 70 Chinese symbols or 140 letters.

From the analysis of the PDU coding style, we can see:

First, in this coding style, all the components shows a rather tight relationship, and all the positions of the segments are defined inside the message itself. For instance, the UDL segment describes the length of the payload and if the real length don't match the UDL, then an error will occur.

Second, the variation of the controlling segment's value will have a deep influence in the whole structure of a message. When the content doesn't match the controlling segment, the receiving process may be in chaos and end with an error. For example, PID consist of only 1 byte, while the different value of PID will have a significant influence on the processing of the payload. When the last 5 bits of PID is "00100", the data will be processed as a voice message, and when it is "00111", the data will be processed as a video message. Obviously, it is quite easy to get into an error with the misunderstanding of the payload.

After the analysis of the PDU code, we determine to produce the abnormal messages by the unmatched length of the payload and UDL. At the same time, we will mutate the controlling variables and produce the final SMS testing samples.

Testing platform

Hardware

For hardware, we mainly rely on the development of SDR(Software Defined Radio), which is a new kind of wireless broadcasting and communication technology.[4] It is not based on the traditional hardware RF design, but the wireless communication protocol defined by the elastic software.

In the system, we will use the RangeNetwork, which is a device integrated with a computer and a SDR board, to emulate the base station's operations on the message. This device is developed for experimental use or construct low power consumption network. First we should install the OpenBTS suit on the computer and after a few simple steps of configuration, we can do some emulations of the SMS service with it.

Software

GNU Radio is an open-source SDR suit scripted mainly by Python. A Python script can configure a RF platform into various kind of sensors. It can detect and analyze signals of vehicles, WIFI, Bluetooth and so on. So it is a very suitable software for project aiming at SDR development. If we install OpenBTS on the basis of GNU Radio, the SDR platform will be configured to be a professional base station tool.

OpenBTS is an interface of GSM based on UNIX. Without the signal of the telephone company, it can provide standard service for those cellphones which are compatible with GSM, use SDR platform to emulate the Air Interface(Um), make connections with the standard 2G device, and use SIP soft switching protocol or PBX to make a call.[5] Recently, the development organization of OpenBTS is trying to add 3G protocols to the software and make it more useful.

Based on these hardware and software, we can build a closed testing system in which we can sending free messages of arbitrary number.

Sample producer and Log system

The testing system consists of an integrated SDR system, a cellphone to be tested and a central computer which produces and sends samples to the SDR. We install the UBUNTU LINUX OS, then GNU RADIO and OpenBTS need to be added.[6] And after all the preparation, we can control the testing system by simple commands.

After a period of time, artificial testing show the disadvantages like more errors and low efficiency. We need to write a script and run the test automatically. Some programs of producing the samples and sending the commands are installed in the central computer to improve the performance of the whole system and that works well.

At the same time, a monitor software which can handle the suspicious messages are essential to the system. We developed an APP on the cellphone which can monitor the whole process of the receiving of the messages. The APP is able to automatically analysis the state of the cellphone, judge if there is anything wrong with the thread of message, and save the monitored data to the log system.

In details, the monitored objects are several variables of the phone which indicate the state of the process, including memory, some key data in the CPU. We need to monitor the message thread. When the thread break down, the samples can be regarded as effective and we can look up to the binary stream, pause the program for a moment and restart in case the following samples in the period of shutting down can not be received properly.

The content of the software in the cellphone includes the following modules:

(a) Surveillance

This part is mainly responsible for the running state of the message process. When the process crash, it will give commands to indicate the controlling center pause the message sending program and wait for the recovery of the message process. Besides, it will notice the Log module that

something is wrong with the message process and ask for making a backup of all the related variables.

(b) Log

Log part is mainly responsible for the recording of the message which make the system break down. When an error occurs, Log part will back up the binary data of the message and look up the variables in memory and CPU so that we can dig up the vulnerabilities which cause the error afterwards.

(c) Communication

COM part keep connections with the sending computer and guarantee the whole testing system run simultaneously. The sending computer and the tested phone communicate with each other by WLAN. Each time the message is sent, the sending part stay quiet and wait for the comfirmation from the receiver. At the same time, the computer will send the binary data of the message to the phone by WLAN so that a comparing with the receiving data can be done.

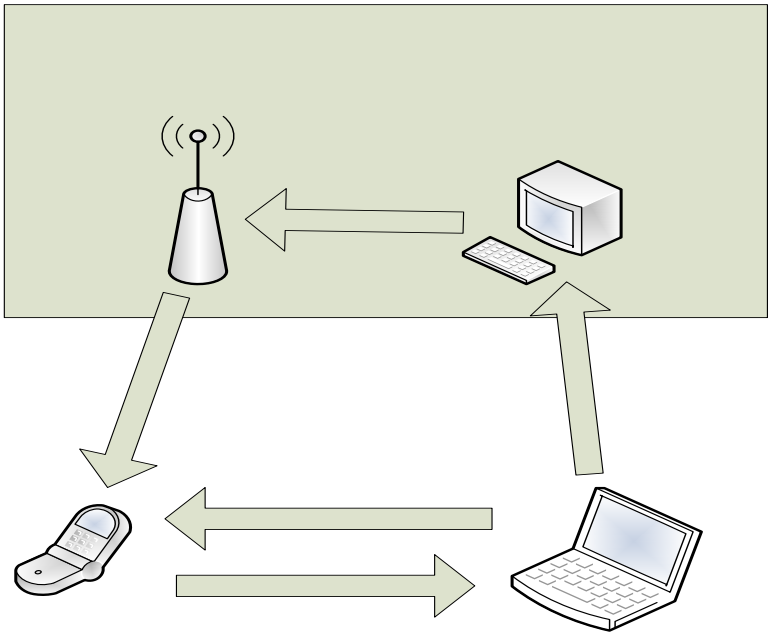

Fig.1 Architecture of the Fuzz testing system

Result

The following picture shows that the cellphone is receiving massive abnormal messages, and some of them will be wrongly analyzed and displayed in chaos.

There are some other interesting discoveries in the experiments. The interval period between the deliveries of messages can influence the result. A rather short interval period may cause the failure of receiving or sending. A proper time interval can be obtained by transceiving massive normal messages before the vulnerability tests. We will choose the least time interval which make the rate of successful transformation under the tolerance rate.

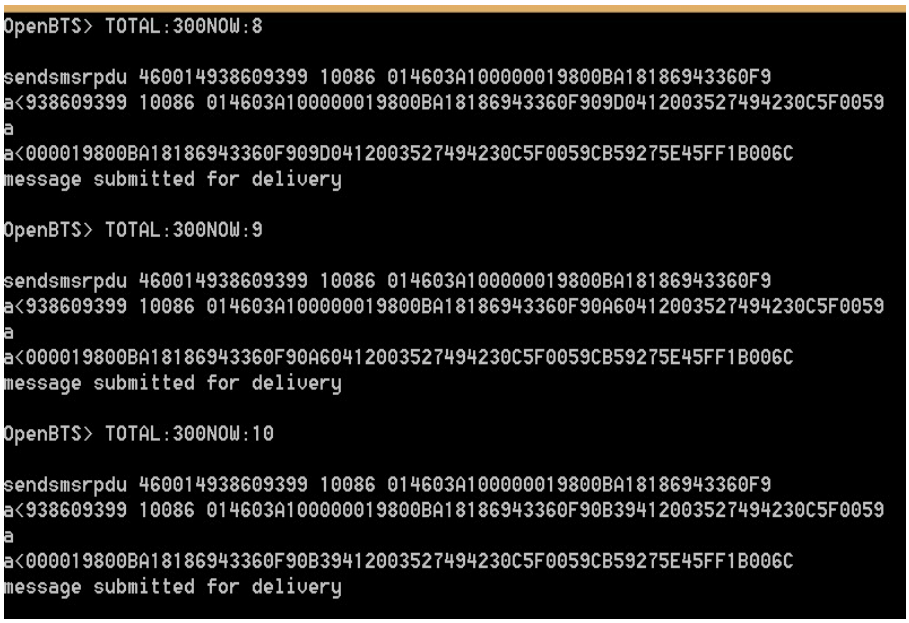

Fig.2 testing samples being sent

Conclusion

In conclusion, this experiment proves that the FUZZ testing method aiming at the SMS module of cellphones is feasible and effective. And the fact that some cellphones can't pass the test indicates that a deeper research in the protocol of message is necessary.

In the article, we build a rather elementary FUZZ system and runs some basic and initiative tests. Except for the normal ones, some samples were filtered directly because of the wrong format, while the others make the message process break down. These phenomenon proves the rightness of the method.

But there are still some problems in this project: we can't figure out what make the process break down due to the lack of knowledge of the source code of the phone, and the algorithm in producing the testing samples is rather initiative. I hope these problems can be properly solved in the future.

References

[1] OpenBTS-4.0-Manual http://openbts.org/

[2] SUTTON M，GREENE A，AMINI P．Fuzzing：brute vulnerabilitydiscovery[M]．[S.1.]：Pearson Education Inc，2007

[3] OEHLERT P．Violating assumption with fuzzing[J]．IEEE Security and Privacy，2005.

[4] MILLER B P，KOSKI D，LEE C P，et a1．Fuzzing revisted：a reexamination of the reliability of UNIX utilities and services[R]．Madison：University of Wisconsin Madison，1995．

[5]ANDREA L，LORENZO M，MATTlA M，et a1．A smart fuzzer for x86 executables[C]//Proc of the 3rd International Workshop on Software Engineering for Secure Systems．[S.1.]：IEEE Computer Society．2007.

[6] VUAGNOUX M．Autodafe：an act of software torture[EB/OL]．(2006)．

http://autodafe.sourceforge.net/docs/autodafe.pdf.