HTML and PDF fuzzing methodology in iOS

Je-Gyeong Jo
Dept. of Computer Engineering
Chungnam National University
Daejeon, South Korea
+82-42-821-7443
oroi@cnu.ac.kr

Jae-cheol Ryou
Dept. of Computer Engineering
Chungnam National University
Daejeon, South Korea
+82-42-821-7443
jcryou@home.cnu.ac.kr

ABSTRACT

iOS is well-known operating system which is strong in security. However, many attacking methods of iOS have recently been published which are called “Masque Attack”, “Null Dereference” and “Italy Hacking Team’s RCS”. Therefore, security and safety is not suitable word to iOS. In addition, many security researchers have a problem to analyze iOS because the iOS is difficult to debug because of closed source. So, we propose a new security testing method for iOS. At first, we perform to fuzz iOS’s web browser called MobileSafari. The MobileSafari is possible to express HTML, PDF and mp4, etc. We perform test abnormal HTML and PDF using our fuzzing method. We hope that our research can be helpful to iOS’s security and safety.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection;

General Terms

Security

Keywords

iOS, HTML, PDF, Fuzzing, Jailbreak, MobileSafari

1. INTRODUCTION

Recently, Mobile market has increased because mobile device has become one of ubiquitous device so it is found easy in our around. With the explosion of mobile device, mobile devices are used not only for private purpose but also for business purpose. However, security of mobile device may not follow to mobile device’s development and mobile market increasing. For example, hacking team of Italy can control iOS device using "Remote Control System (RCS)". Therefore, such release of security issues is a chance that we have consideration to solve.

Most mobile device is using android or iOS for operating system. Android is open source and released by Google, however, iOS is partially open source and released by Apple. So, manufacturer can modify the Android but iOS is only possible to be modified by Apple. Therefore, Android is easy to be modified by many security researchers who perform to analyze android vulnerability. As a result, many researches of android have performed. iOS, however, is difficult to gain information and analyze so iOS needs reverse engineering for analysis. So, many researchers in security avoid analyzing. Therefore, we propose iOS security research to find vulnerability in MobileSafari, and we hope it to be base technology for automation tool for iOS security.

IMCOM '16, January 04-06, 2016, Danang, Viet Nam
© 2016 ACM. ISBN 978-1-4503-4142-4/16/01 $15.00
DOI: http://dx.doi.org/10.1145/2857546.2857555

2. RELATED WORK

2.1 Fuzzing the Phone in your Phone

This research performs by Charlie Miller who is famous in iOS security to research and discuss Android, Windows Mobile and iOS [1]. It tries to fuzz SMS in mobile device. In general, iOS is used CommCenter service that can control WiFi, Bluetooth, 2G and 3G. CommCenter, other service and most app have PLIST file for control and management. In this research, it modifies “DYLD_INSERT_LIBRARIES” property of PLIST file, which works like “LD_PRELOAD” in Linux. The PLIST file can use as library preload attack, so it makes some function and inject to CommCenter. Especially, it can hook “Open” function and get SMS information. Therefore, it can fuzz SMS in mobile device. This research find two vulnerabilities in iOS using SMS fuzzing.

However, this research must operate in jailbreak environment, because this research uses library preload attack to iOS service. So it is different from our research that works without jailbreak environment. In addition, our research has a goal that it finds remote vulnerability in MobileSafari. That is main difference from our research is to find remote vulnerability in SMS.

2.2 Find Your Own iOS Kernel Bug

This research is progressed by Chen Xiaobo and Xu Hao and they perform to fuzz iOS Kernel [2]. iOS and OSX use XNU kernel which are based on BSD, so some code of kernel is opened to public. So, they get kernel information from kernel source is opened. Because of opened kernel source, they can extract kernel I/O signal and structure for fuzzing. They use two types of fuzzing, "Passive fuzz" and "Active fuzz". In case of "Passive Fuzz", it uses “DYLD_INSERT_LIBRARIES” property in PLIST file like Charlie Miller’s research. They developed client program to send specific signal to kernel. On the other hands, "Passive

Fuzz" uses to hooking specific function. This program sends IOKIT signal to kernel and checks crash info. However, this research have a problem, which must operate in jailbreak environment. Without jailbreak, hooking and sending IOKit signal cannot work. In addtion, this research tries to find local vulnerability. Our research, however, has a goal that work without jailbreak environment, and finding remote vulnerability. So, this research is not similar with our research.

2.3 Making iOS MobileSafari Fuzzer and Fuzzing

This research was announced CodeEngn Conference in Korea [3]. Target of this research is similar to our research. This research performs to fuzz MobileSafari through abnormal HTML files. However, method of sending abnormal HTML data to MobileSafari is different from our research. They develop server and client program, and analyze MobileSafari's memory structure to send abnormal data's URL. For this sending, it also operates in jailbreak environment. Without jailbreak environment, there is problem at recursive call, which reloads changed HTML data. So, they call directly related class in jailbreak environment. However, this research must refresh memory address with every update of iOS because it needs to access specific memory address.

Table 1. Compare Table with related work

	Target	**Remote/Local**	**Environment**
Fuzzing the Phone in your Phone	SMS	Remote	Jail-Break
Find Your Own iOS Kernel Bug	Kernel	Local	Jail-Break
Making iOS MobileSafari Fuzzer and Fuzzing	Mobile Safari	Remote	Jail-Break
Our Research	Mobile Safari	Remote	None Jail-Break

These three researches must operate to fuzz in jailbreak environment. On the other hands, our research need not any condition like jailbreak. So, our contribution to fuzz MobileSafari is without jailbreak environment and need not use any client or app. This is main difference to other research. Our scheme can be proved just using HTML but we extend function to fuzz PDF. Because COMEX who are famous in iOS Jailbreak find PDF vulnerability using fuzzing. And they use that vulnerability to jailbreak iOS devices. So, we extend PDF fuzzing and we perform the research of PDF and HTML fuzzing without jailbreak.

3. METHODOLOGY

Most fuzzing programs use debugging tool to check crash. However, iOS does not support debugging tool except own app which is registered to apple developer center. If we want to fuzz default app MobileSafari from iOS, we will not use debugging tool to check crash. So existing researches for MobileSafari fuzzing use debugging tool in jailbreak environment. Current iOS, however, uses memory protection technology so they are not proper recently iOS 6, 7 and 8.

Therefore, we propose to check crash using system monitoring and logging without jailbreak. However, there is problem, which iOS has no way to read automatically abnormal HTML or PDF data such as android or desktop. To solve this problem, we also propose how to read abnormal HTML or PDF data at MobileSafari automatically.

In conclusion, our research proposes five steps for HTML and PDF fuzzing without Jailbreak. The five steps are as below; 1) method of monitoring system status, 2) method of checking crash information, 3) method of create abnormal HTML and PDF data, 4) method of read abnormal HTML and PDF data in iOS, 5) method of checking fuzzing state.

3.1 System monitoring for crash

Most fuzzing program are used debugger to check crash, but iOS does not support use of debugger without jailbreak environment. So, we need to find a way to check crash and get crash information without jailbreak environment. In case of Android, Logcat provides system crash information with CPU register and memory, etc. [4]. So, we propose to use function for system monitoring (syslogd) and logging (CrashReport). We have been try to check crash by using syslogd, but it only shows either crashed or not. In addition, syslogd has another problem that shows too many logs, so there is overhead to check it and it makes our research to be slow by parsing the all logs.

To solve these problems, we use "CrashReport" to check crash. CrashReport in iOS-only software provides crash information files. If we found crash file using CrashReport, we can think our fuzzing tool to make crash through abnormal data. However, Apple does not open to public API of CrashReport and control message of iOS service so we need to solve these problems by using open source library(libimobiledevice) [5].

3.2 Analyze crash information

Getting crash information is possible to call CrashReport. However, CrashReport may create many files like usage status of CPU or memory, etc. So we must filter files which are results of CrashReport and get specific information to check crash. iOS's CrashReport provides crash file which has extension of "ips" and it will be created to "TXT" format. To get information, we have to analyze the files, which are "Crash Time", "Exception Type", "Exception Subtype", "App launch time", "CPU Register" and etc.

If CrashReport creates crash file of system usage like CPU and memory, they do not have "Exception Type", "Exception Subtype", CPU Register's "PC" (Program Counter). If we get three information from crash file, it will be crash file of app or system libraries. Especially, we focus to the "Exception Type". ″Exception Type" is defined by CrashReport and does not open to public. So, we analyze CrashReport by performing reverse engineering. After that, we get twelve "Exception Type". Our fuzzing method can use "EXC_BAD_ACCESS", "EXC_BAD_INSTRUCTION", "EXC_SOFTWARE", "EXC_SYSCALL" and "EXC_MACH_SYSCALL" among twelve types. And there are subtypes under the "Exception Type", but we need to get experience of fuzzing to define useful "Exception Subtype". So we remain the further issue to define sub types.

Information that we extract from crash file saves to database and synchronize with fuzzing log. To synchronize with fuzzing log, we use fuzzing time and "Crash Time" of crash file. We operate to read abnormal data automatically every two seconds, so we make check at the same time or before one second of "Crash Time" (Crash Time – 1). At this sync time, we can get abnormal data, which occurs crash at HTML or PDF. In addition, we can replay crash by using saved information. It is helpful to know crash or exploitable.

3.3 Create abnormal HTML and PDF

There are a lot of method to create abnormal data to fuzz. In general, method to fuzz is smart, mutation and generation [6]. Smart fuzzing changes specific data after it analyzes file structure. Specific data can guess data type like numeric, memory value, address and string value, etc. Mutation fuzzing changes data without analysis. Mutation fuzzing tests from the beginning to the end without analysis. Before we perform this research, we perform android fuzzing and use smart fuzzing method. However, the result of smart fuzzing is slightly better than mutation fuzzing because mobile fuzzing can perform to fuzz multiple devices at one desktop PC. Most mobile device libraries are support multi-connection through device id. iOS also supports multiple connection, so we can test faster than smart fuzzing using multiple device. Fuzzing at desktop has the problem, which needs a lot of desktop for multiple fuzzing, but multiple fuzzing in mobile device only needs one desktop for multiple mobile device. Using many desktops make problems that are merging and processing log, electric power and lots of heat. Mobile device, however, uses lower CPU usage, power and heat than desktop environment. It just needs multiple USB port to perform to fuzz.

To create abnormal HTML, we use standard HTML of W3C (http://www.w3.org). At first, we get HTML tag list from W3C site and real case of HTML tag from portal site. If we use original tag of W3C, it will have problem, which cannot use all property and miss related tag or script. To get the best result in fuzzing, we use various HTML tag extracted from portal site instead of basic tag in the W3C.

To create abnormal PDF, we use "PDF/XML Architecture - Working Samples" file that provides from Adobe company. This file uses most PDF grammars and functions. PDF Standard supports to various scripts and functions to express table, picture and graph, etc. So we think that sample of Adobe company is the best for PDF fuzzing.

We cannot know what data use for crash and effect to PDF libraries, so we use all bytes from 0x00 to 0xff. First, we read all byte in PDF and HTML files, and change each byte 0x00 to 0xff. If the file length is n, minimum case is (n x 255) and maximum case is (255^n).

3.4 Fuzzing HTML and PDF in iOS

Most fuzzing programs in desktop use system function to execute target program. In case of desktop, fuzzing program can use "system", "createprocess" and other functions. In case of android, fuzzing program can use Intent message [4] to execute app. However, iOS does not support function to be executed app by other app or other method except user behavior. In iOS, it can send specific data or file to other app as "share to other app", but this also needs user behavior. Therefore, this is not proper method for our research.

To solve this problem, we use MobileSafari to fuzz both HTML and PDF. In case of HTML fuzzing, MobileSafari supports to express HTML because it is made for expressing HTML document. However, MobileSafari loads to abnormal data created by fuzzing program and reloads automatically. For this operation, we use "Meta" tag of HTML for recursive call in MobileSafari. "Meta" tag, however, should be defined in front of abnormal data. If "Meta" tag is defined after abnormal data, abnormal data may break "Meta" tag.

In case of PDF fuzzing, we use similar method used at HTML. MobileSafari can express PDF by itself, so we analyze MobileSafari in expressing PDF. If there is no option in PDF file, it is just text data in MobileSafari. MobileSafari cannot express PDF directly, which is just to print grammar of PDF file. To express PDF in MobileSafari, fuzzing program sends specific HTTP header to MobileSafari. Standard HTTP header [7] defines "Content-Type" to express other data except HTML. So we define "Content-Type" as "application/pdf" in HTTP header when MobileSafari loads PDF and abnormal data created by fuzzing program. However, this method still has problem. If fuzzing program sends "application/pdf" as "Content-type", the MobileSafari will not process "Meta" tag for recursive call. Because MobileSafari already processes all HTML to PDF. To solve this problem, we create parent page, which uses "Meta" tag for recursive call and "Iframe" to load abnormal PDF. "Iframe" tag calls child page, which can load abnormal PDF through specific HTTP header.

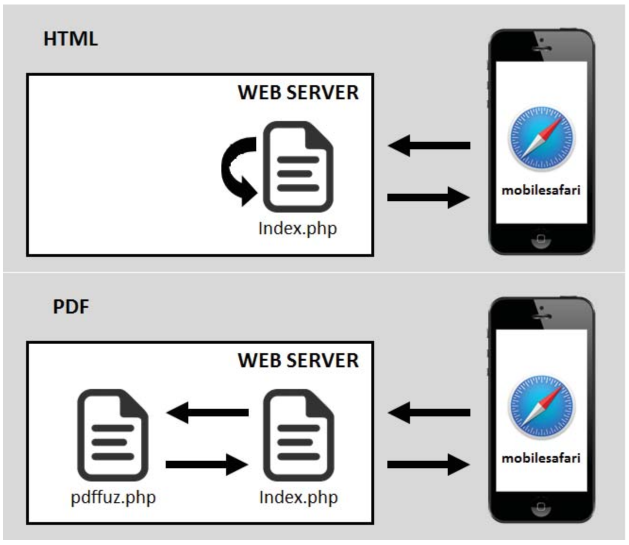
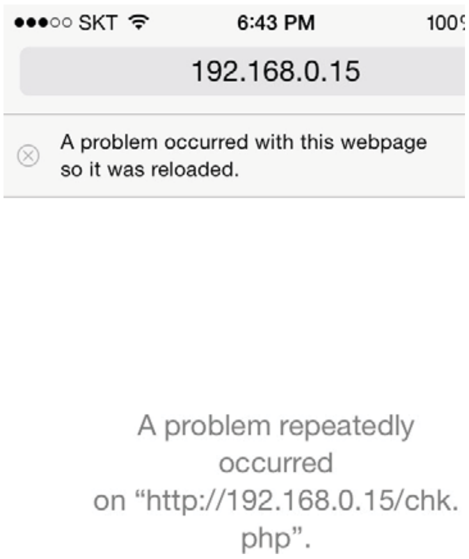


Figure 1 Flow of fuzzing HTML and PDF

3.5 Checking Fuzzing state

In case of HTML fuzzing, we fuzz MobileSafari with HTML, memory leak and CPU usage are not a problem. However, PDF is not default function of browser. When we create abnormal PDF data, MobileSafari notifies several times, which PDF data has problem. So we need monitoring system that fuzzing program works well or not. If fuzzing program does not work well, fuzzing log will not insert to database. So, we need to calculate time difference of last fuzzing log time and current time. If the time difference is more than ten seconds, we will decide that fuzzing program does not work well. At last, fuzzing monitoring system finds term of fuzzing log, it notifies to user by sending e-mail. When administrator of fuzzing program gets error message, administrator can refresh MobileSafari or restart MobileSafari to solve problem.


●●●○○ SKT 6:43 PM 100

192.168.0.15

A problem occurred with this webpage so it was reloaded.

A problem repeatedly occurred on "http://192.168.0.15/chk.php".

Figure 2 Error message when system problem occur

On the other hand, jailbreak device can solve this problem without user action. It can use language like python or C language. So fuzzing monitoring system checks term of log and opens MobileSafari by using command. If MobileSafari gets state of crash or lack of memory, MobileSafari will not refresh the page and create log to database itself. So fuzzing monitoring system needs not to check MobileSafari state, just it checks fuzzing log. If it gets a problem with log, it will execute MobileSafari with URL through "sbutils". "sbutils" is package of CYDIA [8], which is possible to execute MobileSafari with specific URL. "sbutils" has a program called "sbopenurl", which is suitable program we want. Therefore, jailbreak device can solve this problem automatically. On releasing iOS update version, jailbreak program, however, cannot publish so we do not recommend this method.

4. IMPLEMENTATION

Our research uses libimobiledevice [5], PHP and MySQL, etc. for fuzzing program development. iTunes that control iOS device does not support any of function we want. So, we use libimobiledevice's "idevicecrashreport" to get crash information from iOS devices. The code of "idevicecrashreport" is opened to public, so we can modify and add function to "idevicecrashreport". Our program reads crash file through "idevicecrashreport" and extracts "crash time", "Exception Type", "Exception Subtype" and CPU Register's "PC", etc. If our program succeed to extract crash information, it will save this information to database with crash time.

If our program find the same time of fuzzing log, it will save crash log to specific fuzzing log. If our program, however, cannot find same time, it will compare crash time less than 1 second or not. In case of PHP, we use file I/O function to express abnormal HTML and PDF. MySQL uses to save fuzzing log and sync with crash information.

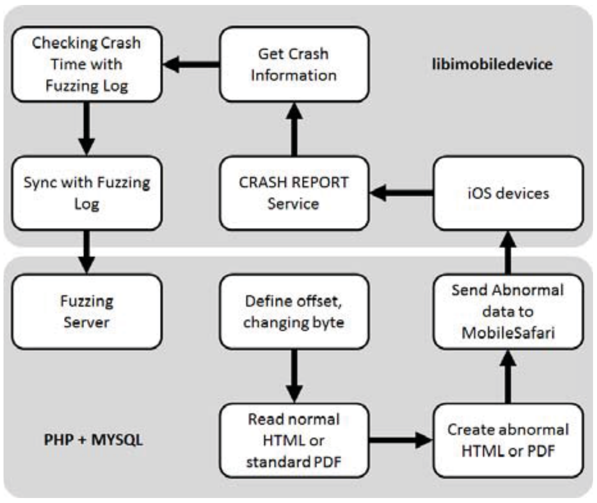
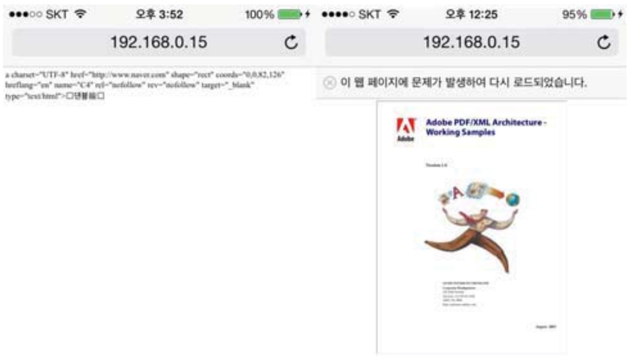

Checking Crash Time with Fuzzing Log
Get Crash Information
libimobiledevice
Sync with Fuzzing Log
CRASH REPORT Service
iOS devices
Fuzzing Server
Define offset, changing byte
Send Abnormal data to MobileSafari
PHP + MYSQL
Read normal HTML or standard PDF
Create abnormal HTML or PDF

Figure 3 Flow of our research

HTML Fuzzing uses recursive call with 2 seconds period, on the other hand, PDF fuzzing uses 3 seconds period. The reason why PDF fuzzing has more 1 second than HTML fuzzing because MobileSafari needs time load PDF libraries. So it takes (length x 0xff) x 2 seconds time to fuzz HTML and (length x 0xff) x 3seconds time to fuzz PDF. However, goal of our research is to fuzz all byte, so we use (255^{length}) x 2 seconds time at least.

If we perform to change from 0x00 byte to 0xff byte, our research will spend more time. Therefore, we need to use more devices to fuzz. We identify the multiple devices by using device id and IP address. PHP cannot check device id remotely, so we must use IP address of each device, which is assigned by router. To use multiple devices, we must separate offset of abnormal data or change byte. If we use number of N devices, we will fuzz N case at one time.

Figure 4 shows our fuzzing program, which prove our method (fuzzing of HTML and PDF in MobileSafari).


●●●○○ SKT 오후 3:52 100%

192.168.0.15

a charset="UTF-8" href="http://www.naver.com" shape="rect" coords="0,0,82,126" hreflang="en" name="C4" rel="nofollow" rev="nofollow" target="_blank" type="text/html">□덴뷁빪□

●●●●○ SKT 오후 12:25 95%

192.168.0.15

이 웹 페이지에 문제가 발생하여 다시 로드되었습니다.

Figure 4 Fuzzing HTML and PDF in MobileSafari

5. CONCLUSION

We progress this research for iOS's security and we want to report that iOS still need to check security. iOS is well known for safety and secure. However, it has a lot of vulnerability. So, we propose automation security checking method in iOS. In the proposed method, we can find a lot of vulnerability that incurs crash. After we analyze crash of fuzzing result, we find vulnerability in iOS 8.3, which is vulnerability of PDF decrypting. However, apple has patched without notice. Therefore, our method prove and is effective to find new or patched vulnerability. We hope that our work can help to iOS security research and automatic security checking in iOS.

After this work, we will try to expand MP4 fuzzing with HTML and PDF and we will test our method to Internet of Things (IoT) devices except iOS because our goal is to analyze security of all devices that is not opened source code or tools to public.

6. ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648)

7. REFERENCES

[1] Collin Mulliner, Charlie Miller. 2009. *Fuzzing the Phone in your Phone*, Black Hat USA 2009

[2] Chen Xiaobo, Xu Hao, 2012. *Find Your Own iOS Kernel Bug*, Power of Community 2012

[3] Nam Daehyeon, 2014, *Making iOS MobileSafari Fuzzer and Fuzzing*, CodeEngn Conference 11

[4] Je-gyeong Jo, Jae-cheol Ryou, 2015, *Method of Fuzzing Document Application Based on Android Devices*, Vol 25, Feb. 2015, 31-37, Journal of The Korea Institute of Information Security & Cryptology

[5] maintained by Martin Szulecki, *Libimobiledevice*, http://www.libimobiledevice.org/

[6] Fuzz Testing, https://en.wikipedia.org/wiki/Fuzz_testing

[7] Header Field Definitions, http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

[8] Isa56, *OpenJailbreak, fuzzyDuck & iOS fuzzing*, Nov. 2013, http://www.isa56k.com/2013/11/openjailbreak-fuzzyduck-ios-fuzzing.html