Randomness of Screen Captured Image for Diverting DroidBox

Dae-boo JEONG, Han-jae YUN and Man-hee LEE*

Dept. of Computer Engineering, Hannam University, Hannam-ro 70, Daeduck-gu, Daejeon, Korea
*Corresponding author

Keywords: Android, DroidBox bypass, Data leak, Steganography, Screen capture.

Abstract. As many people carry their own smart phones and use them for various purposes, hackers’ interests move on toward developing smart phone hacking techniques. Among them, private information stealing becomes one of main topics because the information can be used for subsequent attacks such as smishing. For detecting such smart phone applications, some tainting analysis techniques are proposed to tag important information for tracking the tagged information. One of such evading techniques is to use a screen capture. In this paper, we showed how traditional steganography does not work for this purpose.

Introduction

The smart phone is becoming a main tool for personal and business use. People store almost everything in smart phone from daily memo to very private information such as bank information and social security number. This is why smart phone is now a main target of hackers. In some countries like South Korea where most mobile carriers provide Long Term Evolution (LTE) communications, massive network attack via smart phone is viable. However, due to the low bandwidth provided by mobile communication carriers in most countries, Internet attacks like DDoS are not reported yet. Instead, smart phone hackers are interested in earning money by directly making bogus calls or sending SMS messages to toll phone numbers.

One of lucrative area for hackers is stealing information from smart phone users. As people use smart phones for both personal and business uses, there are so many information that hackers can collect. Personal information includes pictures, bank account number, PIN number, social security number, and so on. Business information depends on how users store in the smart phone. Users can download work files, send and receive e-mails, write up memos for meeting, and so on. As people use the smart phones in wider areas, there is more information interesting to hackers.

Detecting malicious applications that steals information is very hard because there are too many holes to evade such detection techniques. Traditional data leak prevention and detection mechanism is deep packet inspection technique where a system in the middle of network collects all the packets and searches evidences of data stealing. This technique is by-passed by simply encrypting or encoding the information. Furthermore, in the smart phone environment, it is practically impossible to expect mobile communication carriers to operate such systems for many different subscribers. The second data leak detection method is to use digital right management techniques. In the technique, when accessing the information, applications or users need to first acquire appropriate right via right management system installed in the smart phone. This technique is usually applied to multimedia files or some business files, but it is also hard to apply to various information formats.

One of promising and advanced technique is tainting analysis [1,7,8,9,10]. Tainting data is to put a tag to data to be protected and to track the tagged data until the data is destroyed or leaves the system. In the latter case, an alarm is raised for further analysis. Tainting analysis includes tainting and tracking tagged data, and later analyzing how the tagged data is processed before detection. Usually such tainting information is tagged to data down to bit-level granularity, thus requiring at least two times of data storage of smart phones even with one-bit tag information. Furthermore, in machine languages, data moves between registers so this tracking should be supported in hardware. This is
why tainting analysis in real device is technically impossible, so virtual environment is used for this purpose.

One of most famous tainting analysis tools is TaintDroid specialized to perform tainting analysis in Android system [1]. Later, DroidBox, an Android virtual environment developed by Google, contains TaintDroid, so many researchers are using DroidBox for dynamic analysis for malicious apps for Android system [3, 4]. Because of this, hackers and security researchers tried to figure out how evade DroidBox’s tainting analysis while leaking data. Sarwar et al. [2] introduced several approaches not using normal communication paths, but using rather abnormal subjects such as file length, which is generally called subvert channels.

In our previous study [5], we demonstrated how to systemically collect private information via screen capture image. Our approach is to use screen bitmap image where TaintDroid loses its tag information when tagged information is stored on screen bitmap memory. We displayed private information and took a screen shot. Then, we uploaded the captured image to a predestined server. The server receives images from many infected smart phones and utilizes the optical character recognition (OCR) to recognize the texts written on the images.

The main problem of our approach was that we needed to write some texts on the screen. When smart phone users keeps looking at the screen, they can easily figure out something suspicious is going on. In this study, we would like to share our trial and its error to avoid this situation. We thought it would be possible to evade tainting analysis by storing private information to screen image directly and sending the images to a remote server. This paper will first share how and why we failed to do so and also share a hint that we are currently pursue for neat data leaking techniques.

Related Work

TaintDroid and DroidBox

Developed to expand the Android-based smartphone platform, TaintDroid can dynamically analyzes the Android-based apps in the Android emulator environment as a tool for tracking the flow of sensitive data inside smartphones to access apps downloaded from the third party. TaintDroid attaches a tainting tag on sensitive data by modifying the original Android kernel source. Since the tainting tag is propagated to variable or procedure. Simple data access is not detected as a data leak, but, when the data is to be transmitted to the outside via the network or written to the file, it detects that the sensitive data is leaked to the outside. Thus TaintDroid monitors that the sensitive data, such as a unique smartphone data or personal data within the smart phone such as the international mobile subscriber identity (IMSI) and the international mobile equipment identity (IMEI) to be leaked to the outside by using the tainting techniques and writes the results to a monitoring log [1].

DroidBox is a dynamic analysis tool developed using Taintdroid. Droidbox uses the Android emulator on Linux and analyses dynamically of the Android-based app to trace the flow of sensitive data [3, 4].

Side Channel Attacks

Side-channel attack in cryptography is an attack method on physical implementation process to gain hidden information instead of brute force or theoretical attack. Side-channel attack in data leak area is any techniques to steal private or secret information not by sending the information directly, instead by emanating the information indirectly. There are various side-channel attack techniques presented in [2]: Timing Attack, File Length Attack, Bitmap Cache Attack, Text Scaling Attack.

Optical Character Recognition Method

In our previous [5], we used the bitmap cache attack that exploits the fact that DroidBox loses its tracking information when data is stored on bitmap cache for screen display. We developed an application that accesses IMEI and IMSI values, displays the values on the screen, takes a screen capture, and sends the captured image to an outside server. We figured out that recognition rates
differs depending on font sizes and types, but it is quite true that we can acquire private information stably not by being detected by DroidBox. As we mentioned earlier, the main disadvantage of our method is that it is highly probable that users, not DroidBox, can recognize something suspicious is going on because in our method IMEI and IMSI values shows on the screen suddenly. We tried to shorten the timing to display the texts, but it was almost impossible to show texts on screen in a way human eyes cannot detect.

**Randomness of Screen Captured Image**

To overcome the shortcoming of our previous research, we tried to use screen captured image to store secret information. That is, instead of displaying secret text on screen, we tried to embed secret information into an image directly. Then, the image is displayed on the screen before we take a screen shot. If the captured image contains the embedded information after screen capture, we can easily apply general steganography techniques. In this paper, we shared this test result.

In steganography, subtle color information can be modified because human eyes cannot differentiate the difference. The small differences sum up and make some meaningful information. In order to apply this method, we tested as follows. First, we prepared an image and changed one bit value of RGB information of the image. Then, we displayed the image on the screen and captured it. We compared the two images to see whether the changed bit information is preserved.

Interestingly, we can recover the changed information only sometimes; we could not recover the information all the time. We performed many tests by changing bit locations, but we had similar results that we recovered hidden information occasionally.

To see how randomly the values are changing, we compared all pixels’ information. We selected an image from the Internet and embedded IMEI information and displayed on screen and captured it. Figure 1 shows the two images. As we mentioned earlier, they look the same. However, their RGB values are different at some points. To measure the difference, we calculate each pixel’s RGB values and compare all pixels’ values of the two images. The comparison result is shown in Figure 2. We used 145x145 image, so x and y axis has up to 145. Z axis, represented as RGB color difference, is summation of each color value’s difference. For example, when a pixel’s original RGB value (1,1,1) and captured pixel’s value is (0,1,2), RGB color difference is two because the red value is decreased by one and the blue value is increased by one. Figure 2 draws RGB color differences of all pixels. Only 21.2% pixels have zero value, meaning that there is no color difference between the original and the captured RGB value. The average color difference is 2.78. We performed the same test on several images, but we could not find any routine and rules, meaning that pixel locations with the same RGB values kept changing.

![Figure 1. Original (left) and captured(right) images.](image-url)
Summary

Hackers’ main concerns are moving from desktop computer to mobile devices. Among various types of attacks, personal and business information stored in mobile devices becomes the main target. The information can be utilized as back data for personalized spear attacks. This is why it is important to detect malicious mobile applications as early as possible before they cause personal and business damages. To detect data leaking applications, tainting analysis is adopted. Under tainting analysis, once tag information is attached to memory, even to bit-level information, the tagged information keeps follows when other contents of memory move and modified. DroidBox, with very strong tracking capability, can detect many data leak.

Side channel attack applied to data leaking application can make a subvert channel to leak private information without detection by DroidBox. Among many possible attack methods, we utilized bitmap cache attack in which private information is written on screen and captured for later analysis. In our previous research, we directly wrote some private information on screen and applied an OCR method to recover the information. However, users can detect the attack by simply watching the screen. To overcome this disadvantage, this study tried to hide the private information by using general steganography. If the information hidden in the image can be still recovered after image capture, we could simply hide all the information into a single image.

Several experiments showed that captured image is quite different from the original image. Definitely the captured image and the original image look the same to human eyes, but its RGB values are different without any rules. Their RGB value differences and locations of different color values are also random, meaning that it is almost impossible to hide private information to an image directly.

We currently keep investigating into how to hide information into images. Even though we cannot recover information hidden in one image, we could recover some part of information with one image or multiple images. We also keep searching how to leak information from smart phone in a way previously unknown yet.

Acknowledgement

This research is financially supported by 2016 Hannam University Research Fund.

References

[1] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P.Cox, Jeayeon Jung, Patrick McDaniel and Anmol N. Sheth, TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, OSDI vol. 10. pp.255-270, October, 2010.
[2] Golam Sarwar(Babil), Olivier Mehani, Roksana Boreli and Mohamed Ali Kaafar, On the Effectiveness of Dynamic Taint Analysis for Protecting Against Private Information Leaks on Android-based Devices, NICTA Technical Report RT-7091, 2013.

[3] P. Lantz. droidbox - android application sandbox, February 2011.

[4] Google code - DroidBox, https://code.google.com/p/droidbox/

[5] Yeon-Kyung Kim, Han-Jea Yoon, Man-Hee Lee, “Experiment of Side Channel Attack for Diverting Data leak Detection by Taint Analysis of DroidBox,” CISC-S’15, Jun 2015.

[6] https://en.wikipedia.org/wiki/Side-channel_attack

[7] Anastasia Shuba, Anh Le , Minas Gjoka, Janus Varmarken , Simon Langhoff , Athina Markopoulou, Demo: AntMonitor: A System for Mobile Traffic Monitoring and Real-Time Prevention of Privacy Leaks, Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, September 07-11, 2015, Paris, France.

[8] Yihang Song , Urs Hengartner, PrivacyGuard: A VPN-based Platform to Detect Information Leakage on Android Devices, Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, October 12-12, 2015, Denver, Colorado, USA.

[9] Sen Chen , Minhui Xue , Zhushou Tang , Lihua Xu , Haojin Zhu, StormDroid: A Streaminglized Machine Learning-Based System for Detecting Android Malware, Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, May 30-June 03, 2016, Xi'an, China.

[10] Gerardo Canfora , Eric Medvet , Francesco Mercaldo , Corrado Aaron Visaggio, Acquiring and Analyzing App Metrics for Effective Mobile Malware Detection, Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics, March 11-11, 2016, New Orleans, Louisiana, USA.

[11] Xusheng Xiao, Nikolai Tillmann, Manuel Fahndrich, Jonathan De Halleux, Michal Moskal, User-aware privacy control via extended static-information-flow analysis, Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, September 03-07, 2012, Essen, Germany.

[12] Joseph Chan Joo Keng, Tan Kiat Wee, Lingxiao Jiang, Rajesh Krishna Balan, The case for mobile forensics of private data leaks: towards large-scale user-oriented privacy protection, Proceedings of the 4th Asia-Pacific Workshop on Systems, p.1-7, July 29-30, 2013, Singapore, Singapore.