Privacy-Aware Gateway to Prevent Privacy Leaks from Smart Devices

Ahmad Bazzi[1,a)] Yoshikuni Onozato[2,b)] Yuta Kiriyama[3,c)]

Abstract: Smart devices are becoming an integral part of our daily life due to the various applications they can run. One recent study evaluated around 4000 applications and found out that more than one fifth of the evaluated applications leak phone identifiers such as IMEI and phone number [17]. Many users would consider this as an invasion of their privacy; however, they are either unaware of the stealthy activities of certain installed applications or they don't know how to prevent it. Similarly many companies have strict policies regarding leaking any data found on corporate phones and tend to reject most applications for this reason. We propose using a modified privacy-aware gateway that can block privacy leaks from the connected phones. We setup a working installation and get successful results with various applications; any phone identifiers are removed from the sent packets. Using this solution we can improve users' privacy without consuming the computing resources of the smart devices.

1. Introduction

Google encourages mobile software developers to create different kinds of applications for Android devices where application categories range from finance to games. When a user chooses an application and clicks on the install button, he is faced with the set of required "App permissions" that he must accept before downloading and installing the selected application. The user must explicitly accept the required permissions to install an application as shown in Fig. 1. Although some applications don't require any special permissions, many apps require "Network communication: Full network access." Some applications go a step further and require access to "Phone calls: Read phone status and identity." This permission allows the application to access unique identifiers such as [20]:

IMEI International Mobile Equipment Identity
MEID Mobile Equipment IDentifier
ESN Electronic Serial Number
IMSI International Mobile Subscriber Identity

Consequently combining the above two permissions would give an application the ability to send unique identifiers of the smart device through the Internet. More examples can be found in [1].

On the other hand, the requested permissions might be incomprehensible for some users to understand, so they just click the "accept" button. In fact, research by Kelly et al. showed that although permissions displays are read, they are generally not understood which prevents users from making fully-informed decisions [12]. It is also worth noting that in one lab experiment, Kelly et al. found that presenting the privacy permissions in a more comprehensible manner while the user is making her decision, and not after, allowed the users to make choices that better protect their privacy [13].

Moreover, research suggests that the timing of showing the privacy permissions affects the users' decision. In other words, the user might decide differently if she was presented with the required App permissions while making her choice. Hence the user might tend to ignore the required permissions as they appear after the user has already decided to download an application and clicked the "install" button [6].

The required App permissions show one side of the situation, the other side can be understood by considering the "privacy policy" that might be provided by the application vendor. Generally speaking, professional companies tend to provide a privacy policy specifying the user data that their application collects. However, such privacy policies tend to be ignored and left unread by most users. A privacy policy might explicitly state that the application uploads uniquely identifying information of the user to a remote server. Yet, without spending enough time to read the privacy policy, the user would never know what the application does with her data. Even if a user is willing to fully read the privacy policy, the obligation to provide a privacy policy depends on the laws of every country. Hence, several companies and application vendors don't offer any privacy policy and give themselves the right to leak any data they want, and even increase the data they access without even notifying the user — as long as the App permissions have not changed.

Even with the absence of a proper privacy policy, the permission to access the Internet is not necessarily malicious. An ap-

[1] Graduate School of Engineering, Gunma University, 1–5–1 Tenjin-cho, Kiryu, Gunma 376–8515, Japan
[2] Division of Electronics and Informatics, Faculty of Science and Technology, Gunma University, 1–5–1 Tenjin-cho, Kiryu, Gunma 376–8515, Japan
[3] Graduate School of Science and Engineering, Gunma University, 1–5–1 Tenjin-cho, Kiryu, Gunma 376–8515, Japan
[a)] ahmad@nztl.cs.gunma-u.ac.jp
[b)] onozato@cs.gunma-u.ac.jp
[c)] kiriyama@nztl.cs.gunma-u.ac.jp

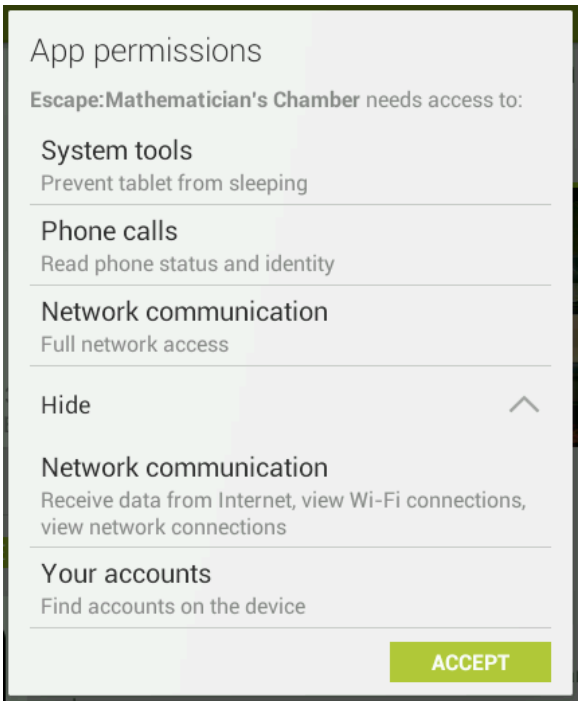

Fig. 1 AppPermissions

plication might need Internet access to retrieve certain data from Internet, such as email messages, maps or search results. It might also need Internet access to show advertisements in the case of free ad-supported applications and ad networks are not always privacy-invasive. In the age of cloud services, there might be numerous other reasons to access the Internet, such as backup to cloud based storage, etc. To know what data an application might collect, a user needs to refer to the privacy policy.

As applications access the Internet, it is generally difficult to know what data are being sent or when. The application might be sending anything from usage statistics to Internet browser history, depending on the access permissions it has. To face these challenges, some automated way is necessary to protect the common user who wants to enjoy and benefit from the available applications without delving into the technical details.

The contribution of this paper can be summarized as follows:

(1) We propose using a proxy gateway server that utilizes Man-in-the-Middle attack as a solution to prevent the leaking of private data from smart devices.
(2) We implement a working prototype to confirm the feasibility of this approach.

Throughout this paper, we focus on Android mobile phones because Android is open source and can be installed on a wide variety of devices, including computer virtual machines. However, the proposed techniques should equally apply to smart devices using other operating systems.

Furthermore, we focus in this paper on the leaking of private data related to smart devices, but the proposed solution can be equally used to prevent the leaking of any type of confidential or private information. One example would be preventing children from exposing their home address or any identifying information on the Internet, not only from a smart device but also from a computer. The general architecture remains the same using a gateway proxy capable of modifying the passing packets.

This paper is organized as follows. After mentioning the current problems with application permissions in this Introduction, we discuss the threats facing smart device users in Section 2. In Section 3, we present the current approaches to face the diverse threats against smart devices and related works and solutions. We follow this by presenting our proposed solution in Section 4. We discuss the implementation of our proposed solution in Section 5 and evaluate its advantages and limitations in Section 6. Finally we write our conclusion in Section 7.

2. Threats

Threats against mobile phones can be divided into three types:

Malware Malware applications include viruses, Trojans and worms. As the name conveys, the intent of such applications is malicious and might include damaging the device, spying and stealing user information, sending premium-rate SMS for the profit of the malware author, joining a botnet among other illegal purposes. By the mid of 2013, Kaspersky Lab has identified more than 100,000 unique malware samples consisting of 629 families [9]. Moreover, mobile malware is now increasing at a rate faster than ever.

Grayware Grayware is defined as "applications that have annoying, undesirable, or undisclosed behavior" [25]. A mobile application in this category might collect different information on the user such as the e-mail address, phone number, device IMEI, etc. It is also possible that its behavior is within the legal bounds through a careful wording of its privacy policy [8].

Personal Spyware These are applications that target clients who wish to spy on other people [8]. Clients range from police forces to spouses, where common objectives include tracking the target's location and spying on his text messages and phone calls.

To better understand the privacy invasion through common general applications, we need to recognize the widespread of applications that leak unique phone identifiers. Rastogi et al. evaluated 3,968 applications from Google Play and identified more than 21% applications that leak phone identifiers, such as IMEI and phone number and more than 5% that leaked the location of the user [17].It is worth mentioning that in an earlier study by The Wall Street Journal, 56 of 101 popular applications for iPhone and Android transmitted uniquely identifying information without users' awareness [24]. Both of these studies targeted general applications. Next, we refer to research regarding malware samples.

Felt et al. analyze the incentives behind 46 malware applications related to Android, iOS and Symbian. The authors notice that 61% of the studied malware samples steal user information [8]. Similarly, after analyzing 1,500 malicious applications, Spreitzenbarth found out that nearly 57% attempt to steal personal information such as IMEI, address book entries, location, etc. [21].

The separation between legitimate applications and malicious ones becomes more complicated when some malware authors steal legitimate applications and repackage them after attaching

malicious payload [27]. By providing the useful functionality of benign applications, it is even trickier to discover them by the average user. As a matter of fact, Zhou and Jiang observed that 86% of their 1260 studied malware samples are legitimate applications repackaged with malicious payloads [28].

One might think that the requested App permissions can reveal the programmer's intentions. However, in some cases, the application author might mistakenly request a permission that he neither needs nor uses. Stevens et al. analyzed around 10,000 free Android applications from popular markets and observed that "the popularity of a permission is strongly associated with its misuse" [23]; in other words, some application authors are inclined to request a permission because it is popular, not because their application really requires it. The misuse of a permission can be a security risk especially if the application is vulnerable.

3. Current Solutions and Related Works

3.1 Permission-Related Solutions

The current literature considers the problem of privacy and grayware from different angles. One approach is to fix issues related to permissions. Because the current Android permission system does not provide unambiguous comprehensive information about the permissions required by the application, Rosen et al. propose a different approach in presenting the necessary permissions so that users can make better informed decisions [19]. To achieve this, first they map API calls to application behavior types to create a knowledge base. Next, they use this knowledge base to create application behavior profiles. As a result, a user can use this generated profile to make an informed decision whether to install and use the application in question.

Using a different approach, Jeon et al. present an application market system that insert instrumentation codes in each application in order to give the users complete control over an application's behavior. Although the proposed system does not require any modification of the Android OS kernel, the user can monitor and control what an application can access [11].

3.2 Client-Side Anti-Malware Solutions

The anti-virus software is another approach especially that some privacy invading applications fall into malware category. Anti-virus technologies include signature detection, heuristic detection and emulation; however, signature detection remains the fundamental technology in commercial solutions. Based on a regularly updated database, a mobile anti-virus application can detect and block malicious applications. The anti-virus company constantly searches for new malicious applications to update its database. However, an anti-virus cannot detect malicious applications that has not been encountered before and included in its database. Moreover, if the application has stated in its privacy policy that it will access and upload certain private data, the application would not be classified as a malicious application or spyware.

The most popular approach to protect one's mobile device is through an anti-virus application. In fact, Benenson et al. observe that 38% of the 506 surveyed Android phone users have an anti-virus scanner installed [3]. The anti-virus might be pre-installed by the manufacturer or the user might have taken steps to install the anti-virus himself.

Venugopal suggests using signature based detection to create a virus scanning system for mobile devices [26]. The main challenge is the limited resources of mobile devices, hence Venugopal proposes different methods to improve memory usage and scanning speeds to conserve the battery power and memory of the mobile device. Bose et al. propose detecting malicious applications using behavioral detection [4]. Behavioral detection relies on observing the run-time behavior of an application, such as API calls and file accesses, and then comparing it against normal or abnormal behavior profile. Using Support Vector Machines (SVM) as a machine learning classifier, they achieve 96% detection accuracy.

Due to the limited resources of mobile devices, a mobile anti-virus applications is not necessarily as efficient as desktop anti-virus solutions. Rastogi et al. tested the effectiveness of current Android anti-malware applications against common malware transformations. They noticed that current solutions fail against common malware evasion and obfuscation techniques [18].

3.3 Server-side Anti-Malware Solutions

As smart devices suffer from limited computing power compared to current computers, some researchers suggest moving the scanning tasks to a cloud service. Jarabek et al. propose using a cloud-based anti-malware system [10]. By moving the resource intensive tasks to the cloud, the user can reserve his device storage, processing power and battery.

As a shift from client anti-virus applications, in February 2012 and to help combat malware in the official Android application market, Google announced a new service codenamed Bouncer that provides automated scanning of Android applications available through Google Play [14]. A scanning service integrated with the application market such as Google's Bouncer [14] would eliminate the urgent need to install an anti-virus application on one's smart device.

Rastogi et al. propose a framework to analyze Android applications using automated dynamic analysis [17]. In order to detect both malware and grayware, the authors use several detection techniques including taint-tracing using TaintDroid [7], sensitive API monitoring and kernel-level system call monitoring. Spreitzenbarth et al. combine automated static and dynamic analyses, in addition to monitoring and logging calls to native APIs, i.e. not Java, in order to automatically analyze Android applications [22].

3.4 Firewall Solutions

Although an anti-virus application is the most widely adopted solution to combat malware, some users rely on a firewall application as well. A firewall allows the user to specify which application can access the Internet and which one cannot. In general, a firewall cannot be installed and used until the Android device has been "rooted." Rooting a device means that the user has "root" access to its OS, i.e. maximum access permissions. Similarly an iOS device needs to be "jailbroken" in order to install a proper firewall. Although this gives the user complete control over the device, it can also be a security risk as some vendor updates would fail to install or the user might avoid them in order

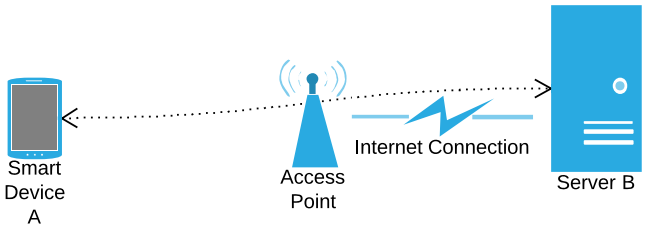

Fig. 2 Smart Device A connects to an access point in order to access the Internet and communicate with Server B.

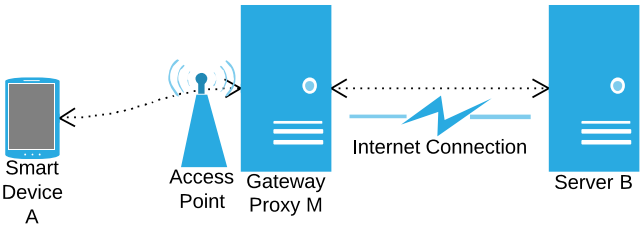

Fig. 3 Smart device connects to the Internet through the Access Point/Gateway Proxy M. Gateway Proxy M is a transparent proxy that performs a Man-in-the-Middle attack and accesses the Internet on A's behalf.

not to lose his privileged access to the system.

There have been recent attempts to create firewall applications that can be setup on Android devices that have not been rooted. This is achieved by creating a dummy tunnel and requiring all traffic to be directed through it. Usually this can give the user the ability to specify which applications can pass through this tunnel and hence access the Internet. Currently, this seems to be the only potential option to use a firewall without rooting one's smart device.

The other limitation is that a firewall would allow or block traffic either based on different criteria, such as the source application and the destination IP address. In other words, if an application needs Internet access permissions for legitimate reasons, it will be allowed and this would give it the chance to abuse this permission. A basic firewall has no way of ensuring that an application is not violating the user's privacy and uploading confidential data.

Nadji et al. use a different approach. Instead of relying on the devices alone, they propose an infrastructure built on the cooperation between network sensors and smart devices. They design and implement a prototype, which automatically identifies malicious traffic through the network sensors and can automatically respond by initiating the proper action via an on-device protected application [15].

In our proposed solution, we rely on specially configured gateway that modifies passing network packets containing identifying information, such as email address, serial number, etc. and replaces these values with dummy information. Moreover, the proposed gateway can be configured to block access to certain malicious servers to help improve the user's privacy.

4. Proposed Solution

We propose using a wireless gateway with transparent proxy that can prevent the leaking of the user's private information. Unlike regular or caching proxy servers, a transparent proxy does not require any client configuration. It mediates certain client requests —such as requests to ports 21 or 80— automatically. It is labeled transparent because the client does not need to know that there is a proxy handling its requests.

Before discussing the network topology with the proposed solution, let's consider the general case of a smart device accessing the Internet through a wireless access point as shown in Fig. 2. The smart device connects to an available access point which will route its requests and sends them over the Internet. Smart Device A can consequently communicate with Server B.

In our proposed solution, a transparent proxy needs to be setup as part of the wireless gateway. In other words, Smart Device A needs to pass through Gateway Proxy M in order to access the Internet and communicate with Server B as shown in Fig. 3. From the client (Smart Device A) point of view, Internet access should appear identical to the case shown in Fig. 2.

This gateway proxy must be able to provide Internet access to the connecting nodes. Moreover, it must be able to prevent the leaking of any confidential information the uniquely identifies the client. This necessitates that Gateway Proxy M intercepts and performs a Man-in-the-Middle (MITM) attack against all the traffic from Smart Device A. Using MITM, Gateway Proxy M can modify the packets before sending them to the target, Server B.

4.1 Man in the Middle Attack

MITM is an attack where one party (M) impersonates another party A as A attempts to communicate with B. There are several variations of this attack depending on the underlying technologies such as networking protocols and authentication mechanisms. In our case A tries to initiate connection with B, but M intercepts this connection as shown in Fig. 3. It appears to A that it is communicating with B directly, when it is in fact communicating with M. M, on the other hand, sends the payloads of the original packets to B, with the ability to make modifications as necessary. Similarly, B thinks that it is communicating with A directly when in fact it is communicating with M. Similarly, M sends the payloads of the original packets to A.

This kind of MITM attack is necessary for Gateway Proxy M to protect A from leaking identifying information. In particular, before resending the packet payloads to B, M needs to scan for any private information and replace them with dummy values. Consequently A won't be able to leak uniquely identifying information; in other words, the smart device won't be able to send its IMEI or serial number, for instance, to B using a clear text channel such as HTTP.

To initiate an encrypted connection over HTTP, A starts by requesting B's certificate. B replies to A with its certificate, and consequently A can use an encrypted connection, HTTPS, to communicate with B. In the case of encrypted connections such as HTTPS, M's role become slightly more complicated. As A requests B's certificate, M must generate a suitable "fake" certificate pretending that it is coming from B and send it to A. Generally speaking, A will consider the certificate as untrusted as it won't be signed by a trusted authority known to A. The target of our experiment is not to conduct an MITM attack, but rather to allow M to monitor and modify A's traffic. Hence, in order to make A accept certificates signed by M, we need to make A trust

the certificate used by M for signing. This can be achieved by installing M's certificate as trusted on Smart Device A. Similarly, A will think that it is communicating directly with B when it is communicating with M, and B will think that it is communicating directly with A when it is communicating with M. Similarly, M should prevent the leaking of A's uniquely identifying information by modifying packets sent to B as necessary.

4.2 Characteristics of the Gateway Proxy

Once M is able to intercept and modify all the traffic from A, both clear text and encrypted, M will be able to prevent certain private data from leaving A. In brief, M must possess the following characteristics:

(1) To prevent privacy leaks, Gateway Proxy M needs to have knowledge of the private information of the users' devices, such as IMEI, phone number, email address, contacts, address book, etc. Consequently M can scan the packets for related privacy leaks and modify the packet payloads as necessary.

(2) The gateway must be able to monitor encrypted connections, in particular SSL-based connections such as HTTPS. This necessitates that the gateway performs MITM attack against all passing connections as mentioned in Section 4.1.

(3) The gateway must be able to replace any private data with dummy data in order to protect the privacy of the connecting user. In other words, M will be resending all packets with new IP and TCP headers and occasionally modified payloads.

(4) For added security, M must be able to block connections, at least to IP addresses that are known to be malicious. M can configured to firewall all connections to black listed servers.

The exact security/privacy policy would depend on the user or the company. In our experiments, we decided that our aim is to achieve the following results:

(1) Allow all data to be sent to trusted servers.

(2) Prevent information leaking by blocking packets containing private data from being sent to servers that have are not in the trusted listed.

(3) Prevent applications from accessing any server on the "black list" and block all packets destined to them.

This is summarized in Table 1.

Table 1 Example rules to be enforced by the gateway proxy

Source	Rule
Trusted Servers	Allow all traffic
Unknown	Block packets with private or confidential information
Known Malicious	Block all traffic

5. Solution Implementation

We wanted to test the feasibility of our solution so we implemented Gateway Proxy M using Mallory, a transparent TCP and UDP proxy [2]. Mallory is originally created to aid in mobile application testing. It runs as a transparent proxy server with MITM capabilities to intercept traffic and modify them in real-time.

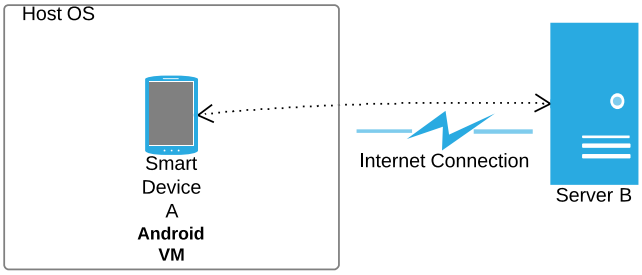

Fig. 4 Smart Device A is setup as an Android VM. It does not notice any transparent proxy intercepting its traffic and appears to be accessing the Internet normally.

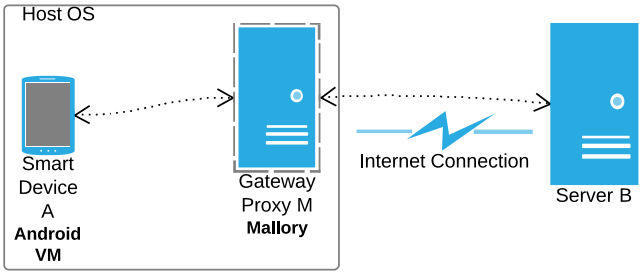

Fig. 5 Smart Device A is setup as an Android VM, and Mallory is setup on a Debian GNU/Linux VM. Android VM is connected to Mallory VM through VirtualBox's Internal Network. Mallory VM is also connected to the Internet using NAT (or Bridged NIC) via VirtualBox. Android VM can access the Internet only through Mallory VM. Mallory VM will in turn perform MITM on the passing traffic.

Mallory can be setup on a notebook —or PC with wireless network cared— to function as a wireless gateway. However, to facilitate testing and experimentation, we used virtualization and setup Mallory as a virtual machine (VM). Using VirtualBox [16] on a Windows 7 PC, we setup two virtual machines:

- Android 4.1.1 VM
- Debian GNU/Linux VM that runs Mallory

It appears to the smart device (Android VM) that it is accessing the Internet directly as shown in Fig. 4. However, all traffic must pass through Mallory VM (Gateway Proxy M) in order for the traffic to be intercepted. Consequently, we configured Mallory VM with two virtual network cards, one card is connected to the Internet through a *Bridged Adapter* or *NAT* (Network Address Translation) provided by VirtualBox. The other adapter is connected to VirtualBox's *Internal Network* like the Android VM so that they can communicate with each other. All traffic from Android VM will pass through Mallory VM in order to access the Internet as shown in Fig. 5. Hence, Mallory VM functions as the proxy gateway to the Android VM.

In order to allow the MITM capability provided by Mallory to function smoothly, we imported the certificate used by Mallory into our Android VM and trusted it. This way the applications will not issue any warning or fail to initiate an encrypted connection based on an untrusted certificate authority.

We looked up the IMEI and serial number of our Android VM and created the necessary rules on Mallory to replace these private data with dummy values. By monitoring the traffic, it was clear that any application trying to leak private information was rendered unsuccessful due to the setup of the gateway proxy.

As an example, consider Fig. 6 which shows TCP

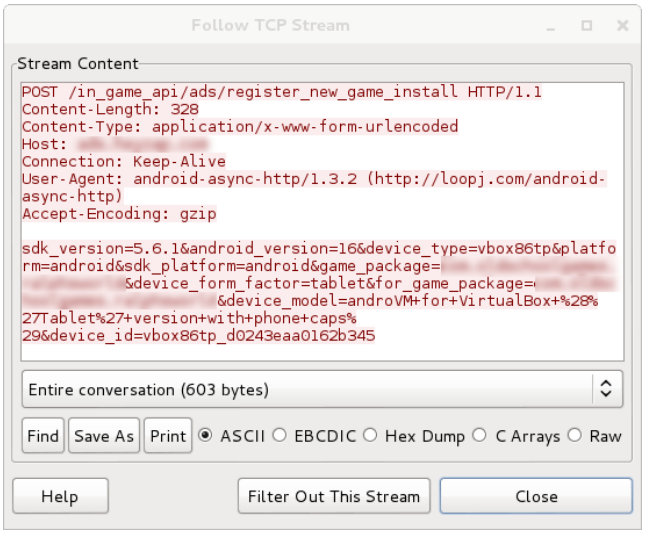

Fig. 6 Example TCP stream that leaks the device ID to an advertisement server

stream viewed in Wireshark[*1]. At the end of this stream, we notice how this application posted the device ID *vbox86tp_d0243eaa0162b345* to a remote server. By configuring the gateway proxy to replace the device ID, we prevent the application from tracking the user. Preventing such leaks is feasible provided that a list of all the identifying details of the related smart devices is prepared in advance.

6. Solution Evaluation

The proposed solution helps prevent applications from leaking private information to untrusted servers and from uniquely identifying the user based on IMEI, email address and similar data. However, for the system to work, the users are expected to trust the gateway proxy and its administrators as much as they would trust the remote server they are communicating with. Hence, we expect this solution to be useful in the following two scenarios:

Corporate Network Companies are very careful about data leaking, such as employees' business contacts. In such a network, the corporate employees have to trust the administrators of this gateway proxy. Although this might be unfavorable for some employees, trusting the company IT administrators is unavoidable. For example, most email servers give the system administrators in charge the ability to access and read the employees' emails; however, ethics and code of conduct would prevent them from abusing their privileges. Likewise, although the proposed solution requires an additional level of trust, we think that it is still feasible to use.

Home Network A home user might implement such a solution for his personal use and possibly share it with other family members. Similarly, trusting the administrator of the device with one's private information is a required condition.

This solution cannot be used when trust cannot be established, such as public spots.

[*1] http://www.wireshark.org/

6.1 Advantages

- Our proposed solution can successfully block all privacy leaks that are sent in clear text and encrypted format over HTTP and HTTPS respectively.
- The firewall can efficiently block access to chosen IP addresses. Alternatively, it can be configured to block access to all servers except specific exceptions.
- The firewall, along with the privacy leaking prevention component, runs on a separate machine and therefore does not consume the limited resources of the smart device.
- An ideal usage for this system includes a company that aims to prevent the employees' devices from leaking private or confidential information.

6.2 Limitations

- The gateway intercepts all encrypted traffic and hence it can gain access to a variety of secret data, such as login usernames and passwords of the users. As mentioned, this necessitates that the users trust the gateway and the party administering it. This solution can protect users' privacy only if they can trust it and its administrator(s).
- For successful MITM attacks, users must trust the certificate used by the proxy. Currently, the most efficient way would be by importing the certificate into their devices.
- As already mentioned, this solution is not suitable, in its current form, for general spots: one reason is that users' passwords would be exposed, another reason is that a manual installation of a fake certificate is necessary.
- Users cannot change system settings. If a user needs to send private information, such as phone number or e-mail address, to an untrusted server, the gateway administrator needs to make the required changes. In a corporate network, a privacy policy might be necessary to minimize such cases.
- The proposed solution blocks privacy leaks over WiFi networks; however, it cannot prevent leaks over other networks, such as Internet over 3G or EDGE. Blocking information over cellular networks would require a modified femtocell [5].
- The proposed solution can intercept encrypted connections using standard SSL; however, it cannot intercept or read data encrypted using a different technique, such as DES, before sending.

7. Conclusion

In this paper we proposed a method to prevent smart devices from leaking uniquely identifying information by using a specially configured proxy gateway. The proxy gateway needs to launch MITM attack against all traffic in order to monitor and modify network packets in real-time as deemed necessary. The aim is to replace the private data —such as IMEI, serial number, phone number, etc.— with dummy data to protect the privacy of the user. We setup a prototype using Mallory proxy on a Linux gateway and confirmed the feasibility of this solution.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us improve this paper.

References

[1] Ahmed, N.: Use Permissions to Secure Your Private Data from Android Apps, (online), available from ⟨http://techpp.com/2010/07/30/android-apps-permissions-secure-private-data/⟩ (2010).

[2] Allen, J. and Umadas, R.: Network Stream Debugging with Mallory (2010).

[3] Benenson, Z., Gassmann, F. and Reinfelder, L.: Android and iOS Users' Differences concerning Security and Privacy, *CHI '13 Extended Abstracts on Human Factors in Computing Systems - CHI EA '13*, New York, New York, USA, ACM Press, pp. 817–822 (online), DOI: 10.1145/2468356.2468502 (2013).

[4] Bose, A., Hu, X., Shin, K. G. and Park, T.: Behavioral Detection of Malware on Mobile Handsets, *Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services - MobiSys '08*, New York, New York, USA, ACM Press, pp. 225–238 (online), DOI: 10.1145/1378600.1378626 (2008).

[5] Davidoff, S., Harrison, D., Price, R., Fretheim, S.: Do-It-Yourself Cellular Intrusion Detection System, LMG Security, pp. 1–77 (online), available from ⟨http://lmgsecurity.com/whitepapers/DIY-Cellular-IDS_2013-08-01.pdf⟩ (2013).

[6] Egelman, S., Tsai, J., Cranor, L. F. and Acquisti, A.: Timing Is Everything? The Effects of Timing and Placement of Online Privacy Indicators, *Proceedings of the 27th International Conference on Human Factors in Computing Systems - CHI 09*, CHI '09, New York, New York, USA, ACM Press, pp. 319–328 (online), DOI: 10.1145/1518701.1518752 (2009).

[7] Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P. and Sheth, A. N.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones., *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI '10*, Vancouver, BC, Canada, (online), available from ⟨http://static.usenix.org/events/osdi10/tech/full_papers/Enck.pdf⟩ (2010).

[8] Felt, A. P., Finifter, M., Chin, E., Hanna, S. and Wagner, D.: A Survey of Mobile Malware in the Wild, *Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '11*, New York, New York, USA, ACM Press, pp. 3–14 (online), DOI: 10.1145/2046614.2046618 (2011).

[9] Funk, C. and Maslennikov, D.: IT Threat Evolution: Q2 2013, Technical report, Kaspersky Lab (2013).

[10] Jarabek, C., Barrera, D. and Aycock, J.: ThinAV: Truly Lightweight Mobile Cloud-based Anti-malware, *Proceedings of the 28th Annual Computer Security Applications Conference - ACSAC '12*, New York, New York, USA, ACM Press, pp. 209–218 (online), DOI: 10.1145/2420950.2420983 (2012).

[11] Jeon, C., Kim, W., Kim, B. and Cho, Y.: Enhancing Security Enforcement on Unmodified Android, *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, New York, New York, USA, ACM Press, pp. 1655–1656 (online), DOI: 10.1145/2480362.2480672 (2013).

[12] Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N. and Wetherall, D.: A Conundrum of Permissions: Installing Applications on an Android Smartphone, *Financial Cryptography and Data Security* (Blyth, J., Dietrich, S. and Camp, L., eds.), Vol. 7398, Springer Berlin Heidelberg, pp. 68–79 (online), DOI: 10.1007/978-3-642-34638-5_6 (2012).

[13] Kelley, P. G., Cranor, L. F. and Sadeh, N.: Privacy as Part of the App Decision-Making Process, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, New York, New York, USA, ACM Press, pp. 3393–3402 (online), DOI: 10.1145/2470654.2466466 (2013).

[14] Lockheimer, H.: Android and Security (2012).

[15] Nadji, Y., Giffin, J. and Traynor, P.: Automated Remote Repair for Mobile Malware, *Proceedings of the 27th Annual Computer Security Applications Conference - ACSAC '11*, New York, New York, USA, ACM Press, pp. 413–422 (online), DOI: 10.1145/2076732.2076791 (2011).

[16] Oracle Corporation: Oracle VirtualBox, (online), available from ⟨http://www.virtualbox.org/⟩.

[17] Rastogi, V., Chen, Y. and Enck, W.: AppsPlayground: Automatic Security Analysis of Smartphone Applications, *Proceedings of the Third ACM Conference on Data and Application Security and Privacy - CODASPY '13*, New York, New York, USA, ACM Press, pp. 209–220 (online), DOI: 10.1145/2435349.2435379 (2013).

[18] Rastogi, V., Chen, Y. and Jiang, X.: DroidChameleon: Evaluating Android Anti-malware against Transformation Attacks, *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security - ASIA CCS '13*, New York, New York, USA, ACM Press, pp. 329–334 (online), DOI: 10.1145/2484313.2484355 (2013).

[19] Rosen, S., Qian, Z. and Mao, Z. M.: AppProfiler: A Flexible Method of Exposing Privacy-Related Behavior in Android Applications to End Users, *Proceedings of the Third ACM Conference on Data and Application Security and Privacy - CODASPY '13*, New York, New York, USA, ACM Press, pp. 221–232 (online), DOI: 10.1145/2435349.2435380 (2013).

[20] Samsung: How to retrieve the Device Unique ID from android device, (online), available from ⟨http://developer.samsung.com/android/technical-docs/How-to-retrieve-the-Device-Unique-ID-from-android-device⟩ (2011).

[21] Spreitzenbarth, M.: The Evil Inside a Droid Android Malware: Past, Present and Future, *First International Baltic Conference on Network Security & Forensics*, pp. 41–65 (online), available from ⟨http://basoti.uni-rostock.de/fileadmin/becker/basoti/basoti12/Tagungsband_BaSoTI2012_Inhalt.pdf⟩ (2012).

[22] Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T. and Hoffmann, J.: Mobile-Sandbox: Having a Deeper Look into Android Applications, *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, New York, New York, USA, ACM Press, pp. 1808–1815 (online), DOI: 10.1145/2480362.2480701 (2013).

[23] Stevens, R., Ganz, J., Filkov, V., Devanbu, P. and Chen, H.: Asking for (and about) Permissions Used by Android Apps, *Proceedings of the 10th Working Conference on Mining Software Repositories*, San Francisco, CA, USA, IEEE Press, pp. 31–40 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2487085.2487093⟩ (2013).

[24] Thurm, S. and Kane, Y. I.: Your Apps Are Watching You: A WSJ Investigation finds that iPhone and Android apps are breaching the privacy of smartphone users, *The Wall Street Journal*, (online), available from ⟨http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html⟩ (2010).

[25] Trend Micro: Generic Grayware, (online), available from ⟨http://about-threats.trendmicro.com/us/archive/grayware/GENERIC_GRAYWARE⟩ (2007).

[26] Venugopal, D.: An Efficient Signature Representation and Matching Method for Mobile Devices, *Proceedings of the 2nd Annual International Workshop on Wireless Internet - WICON '06*, New York, New York, USA, ACM Press, (online), DOI: 10.1145/1234161.1234177 (2006).

[27] Zhou, W., Zhou, Y., Grace, M., Jiang, X. and Zou, S.: Fast, Scalable Detection of "Piggybacked" Mobile Applications, *Proceedings of the Third ACM Conference on Data and Application Security and Privacy - CODASPY '13*, New York, New York, USA, ACM Press, pp. 185–195 (online), DOI: 10.1145/2435349.2435377 (2013).

[28] Zhou, Y. and Jiang, X.: Dissecting Android Malware: Characterization and Evolution, *2012 IEEE Symposium on Security and Privacy*, IEEE, pp. 95–109 (online), DOI: 10.1109/SP.2012.16 (2012).