A Measurement and Security Analysis of SSL/TLS Deployment in Mobile Applications

Yu Guo[1], Zigang Cao[1](✉), Weiyong Yang[2], and Gang Xiong[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
caozigang@iie.ac.cn
[2] NARI Group Corporation, Nanjing, China

Abstract. Secure Socket Layer (SSL) and Transport Layer Security (TLS) have been widely used to provide security in communications. With the rapid development of mobile Internet, they are progressively applied in mobile applications. It is interesting to study the security of their usage. However, most of existed researches on SSL/TLS focus on the whole ecosystem, while few of them have in-depth study on the status quo of mobile security about SSL/TLS. In this paper, we measure the network behaviors of top 50 popular applications on Android and iOS platforms to reveal the security problems of SSL/TLS deployment in mobile Internet. A system is implemented which can extract the handshake parameters and inspect SSL deployment status. We also demonstrate some typical severe problems by performing man-in-the-middle (MITM) attacks against six applications. We believe our study is very consequential for SSL deployment on mobile platforms and the design of secure applications in the future.

Keywords: SSL · TLS · Mobile application security · Measurement · Android · iOS

1 Introduction

According to the report [1] published by China Internet Network Information Center (CNNIC) in December 2015, the amount of Chinese Internet users has reached 688 million and mobile Internet users account for 90.1% of it, which leads to a ubiquitous usage of mobile applications in people's daily life. However, almost every application has access to users' private information, and the problems raised by applications leaking users' sensitive data occur frequently. Therefore, it is legitimate to assess the security of mobile applications and improve it.

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are widely used to provide end-to-end communication security for mobile applications. Although SSL/TLS protocols can be effectively against many types of network attacks, there are still various security problems due to unrated deployment of SSL/TLS in applications. As were shown in previous works [2], some mobile applications leak users' personally identification information (PII), while some applications suffer from the risk of being attacked by MITM [3].

Q. Chen et al. (Eds.): ChinaCom 2016, Part I, LNICST 209, pp. 189–199, 2018.
DOI: 10.1007/978-3-319-66625-9_19

According to the previous works [2–4] on the usage of SSL/TLS protocols, we summarize three main factors that can jeopardize applications into insecure situations:

(1) Improper SSL/TLS deployment on an application.
(2) The usage of low-security parameters in SSL handshake, such as protocol version, cipher suites and so on. Early SSL/TLS versions have protocol flaws, and some of the algorithms used by SSL/TLS are weak, like DES or MD5 algorithms. In addition, the use of self-signed certificates results in risks.
(3) Incomplete certificate validation.

Considering that Android and iOS now account for the major share of the global mobile operating system market, we perform our measurement on Android and iOS mobile platforms. We choose the popular top 50 applications on each platform. We get the dataset via combining manual and automated methods, capturing each application's packets while it is used. Then we design and implement a system that can analyze these packets, detecting whether an application deploys SSL/TLS protocols, and extracting its SSL handshake parameters as well as certificate chains. Apart from the statistics and analysis, we also perform MITM experiments against applications using self-signed certificates and several representative applications.

In summary, this paper makes the following contributions:

- We demonstrate that sensitive data can be accessed easily in some applications, emphasizing the great significance of deploying SSL/TLS properly in mobile applications. We find that there are even a few applications leaking passwords over plain text.
- We design and implement a system for detecting the status of SSL deployment in applications and its security.
- We make statistics on the selection of SSL handshake parameters. We find that 14% of Android apps and 24% of iOS apps use a set of low-security parameters, which is vulnerable when facing attacks.
- We discover some certificate validation problems in several applications. Applications with self-signed certificates fail to resistant MITM attacks.
- Based on our results, we propose concrete recommendations on SSL deployment for application designers to provide more secure applications.

The remainder of this paper is structured as follows: in Sect. 2, we discuss related work. Section 3 explains the origin of our dataset and the methodology of our measurement. Our system is presented in this section as well. In Sect. 4, we illustrate our measurement results, propose concrete recommendations for SSL deployment and describe our future work. Finally, we conclude our study in Sect. 5.

2 Related Work

There have been a few previous works studying the ecosystem of SSL/TLS protocols and analyzing the improper usage of SSL/TLS. He et al. [4] designed a system for vetting the improper SSL usage which may cause vulnerabilities. Their system detects the source code to find design flaws of applications. Sounthiraraj et al. [5] also found

SSL validation vulnerabilities in the source code from Android applications. Thus, the major difference between their work and ours is that we focus on the network traffic generated by applications instead of the source code.

As for the disclosure of sensitive information, recent studies [2, 6] show that when using applications, the user is tracked by third parties. At the same time, the applications being used is leaking users' personally identifiable information without the users' knowledge. Balebako et al. [6] performed a study to reveal and control privacy leaks in mobile network traffic. Roesner et al. [7] presented a method to detect and resistant privacy leaks by the third party trackers. Egele et al. [8] detected privacy leaks in iOS applications. We also have concerns about privacy leaks from mobile applications. However, in this paper, we only discuss the privacy leaks resulted from improper SSL deployment.

Many works in the past have studied the correlation between the SSL handshake parameters and the SSL/TLS server's security. Levillain et al. [9] assessed the quality of https server by analyzing parameters included in the server's response after sending a stimuli to the server. He proposed criteria for assessing TLS quality, which consist of the protocol version, cipher suite, TLS extension, quality of certificate chain and so on. Pukkawanna et al. [10] conducted a research to classify the SSL servers into different security levels using SSL handshake parameters. We reference their understanding of SSL handshake parameters.

Pukkawanna et al. also proposed certificate CA to be an important measure to assess the security of SSL/TLS servers [10]. A certificate issued by a CA offering low-price or free certificates is likely to be a risky certificate. Besides, Georgiew et al. [11] tried MITM attacks against several applications and found over twenty certificate and hostname verification vulnerabilities. Trummer and Dalvi [3] used BurpSuite to detect applications' certificate validation. They found that 43 applications failed to validate trust chain and 59 applications failed to validate hostname matched. We also attempted MITM attack against applications using self-signed certificates, to reveal the security problems existing in mobile applications.

3 Measurement Method

In this paper, we focus on the mobile application security in terms of SSL/TLS protocols. This section describes the data sources and how we performed our measurement in detail.

3.1 Data Collection

Considering the better representation of all applications, we choose the top 50 applications on two major operating systems, iOS and Android. We use a spider to obtain the popular top applications list in the apple app store on Nov 12, 2015. The list contains applications' name, ranking, version, etc. We try to obtain the Android top popular application list at the same time. Because Google has no cooperation with the Chinese mainland, application rankings in Google play cannot represent the use of applications in China. So we select three main application markets in China, namely

360 mobile assistant, Tencent yingyongbao and Baidu mobile assistant. Then we combine the application rankings of these three markets and choose the top 50 popular applications on Android platform.

The basic method of obtaining the dataset is to capture packets when an application is used. First, we set up a wireless environment with 360 portable Wi-Fi. Then when the experimental device connects to the wireless, we capture the traffic flowing through the virtual wireless network card, which is generated by the device.

It should be noted that we not only capture the entire traffic of applications, but also keep the specific traffic of "critical behaviors" for further analysis. We define critical behaviors as operations involving or probably involving users' sensitive information when using applications. We list all critical behaviors used in our measurement in Table 1.

Table 1. List of critical behaviors.

Critical behavior	Sensitive information it may involve
Register	User account, password, phone number, email address, user's personal information
Login	User account, password, phone number, email address
Modify password	Old password, new password, phone number, email address
Pay	User account, payment password
Upload/Download file	Document content
Chat on line	Conversation content
Send/Receive emails	Senders and recipients' email address, content of emails

For applications on iOS platform, we capture packets of each application manually. The experiment device is an apple iPad air1 and its system is iOS 9.1. We try almost all functions of an application and capture packets with Wireshark and Commview.

For applications on Android device, we achieve a semi-automatic capture. We develop an application named Simulator. It consists of a client on the computer and a server on an Android device. We root a Nexus 5 with Android 5.0 system and install Simulator's server on it. Simulator can help us capture traffic generated by the device and send packets to the computer to save automatically. The only thing we have to do manually is to click the figures corresponding to the functions of each application.

3.2 Measurement System

After the data collection process, we get 1000 ".pcap" files (2000000 packets). According to the factors we proposed in Sect. 1, we wish to know how an application deploys SSL/TLS and what parameters its server selects during the SSL handshake. To address this problem, we design and implement a system.

Figure 1 presents the architecture of the system. In the pcap platform we put packets which were saved as ".pcap" format into it. These packets will be replayed and directed to a network card. Then the traffic flows into the TCP parser module. TCP parser module reassembles the TCP segments and restores the message. When the complete message is sent to the next module, the SSL parser module will recognize SSL traffic and discard non-SSL traffic. After that, SSL traffic is further parsed. It will extract parameters of three phases, Client Hello, Server Hello and Certificate. Parameters in Client Hello include client protocol version, host, cipher suites, compression methods, extensions and so on. So does this in server hello. In certificate phase, the module extracts the length of certificates, algorithm, issuer, subject and the validity. All these parameters are saved in "JSON" format and uniquely identified by the four-tuple (source IP address, destination IP address, source port and destination port). Moreover, the module extracts the corresponding certificate chain. The JSON files will be further processed and put into MongoDB database automatically. Finally, the analysis module does statistics and gives a preliminary statistical result.

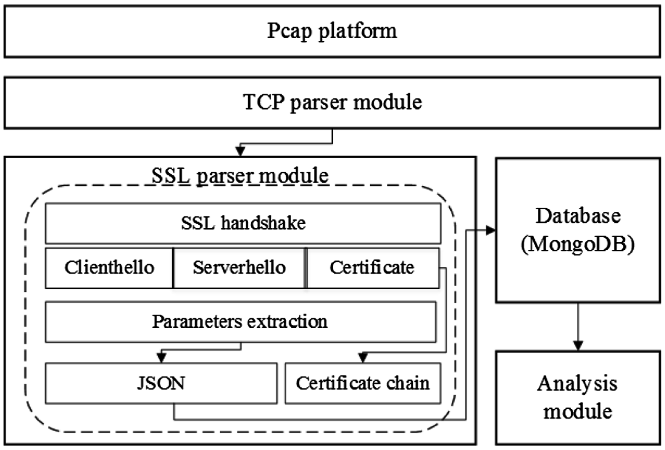

Fig. 1. System *Architecture.* Put the dataset into Pcap platform, then the data is parsed by TCP parser module and SSL parser module sequentially, generating JSON format files. These files are imported into the Database and analyzed by the Analysis module in the end.

3.3 MITM Experiment

As mentioned in Sect. 3.2, besides the parameters of Client Hello and Server Hello, we also extract certificate details and certificate chains. Generally, if an application uses a validated certificate issued by a well-known certificate authority (CA) for the server side, and the client side performs a correct certificate validation, it can protect users from man-in-the-middle (MITM) attack. Otherwise, it provides opportunities to MITM attacks. We perform MITM attack experiments to applications whose certificates are self-signed since they are probably vulnerable to MITM attacks.

We exploit SSLsplit [12], an open-source tool for MITM attacks against SSL/TLS encrypted network connections, to do the experiment. We use 360 portable Wi-Fi to build a wireless network and configure SSLsplit on Ubuntu. As described in Fig. 2, when the mobile device connects to the wireless, the traffic is intercepted by SSsplit. We keep the logs of that using a python script. If there is a non-empty log about 443 TCP ports and the applications on the device can be used normally, that means SSL traffic is decrypted and the communication has been compromised. In contrast, if there is no log about 443 TCP port or the size of 443 port log is zero, the application can resistant MITM attacks.

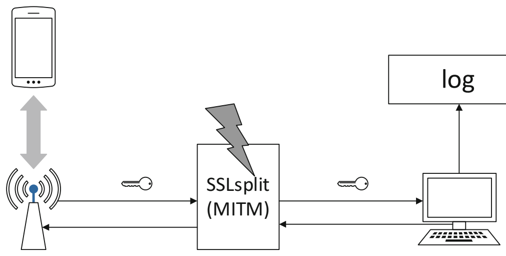

Fig. 2. MITM experiment *deployment diagram*. The wireless is built by 360 portable Wi-Fi plugged on the computer, on which SSLsplit is configured as well.

4 Measurement Results

In this section, the measurement results are presented and analyzed, as well as the discoveries from the MITM experiment. Meanwhile, some concrete recommendations on SSL deployment are proposed for application designers. Finally, we summarize our research and describe the future work.

4.1 SSL/TLS Deployment on Applications

Through the measurement, we discover that the SSL deployment status on applications can be classified into three types:

a. Deploy SSL/TLS protocols on all critical behaviors. *We name this type "all".*
b. Only deploy SSL/TLS protocols on some of the critical behaviors. *We name this type "part".*
c. Not deploy SSL/TLS protocols on any critical behaviors. This type includes two different cases. If an application uses proprietary protocols to encrypt communications, it is relatively secure although it does not deploy SSL/TLS protocols. However, applications using http or other protocols on all critical behaviors may have serious security problems. *We name these two cases "none-security" and "none-insecurity" separately.*

Among top 50 applications on iOS platform, 28 applications deploy SSL/TLS protocols on all critical behaviors. Three applications only deploy SSL/TLS on part of critical behaviors, still exposing sensitive data on non-SSL/TLS deployment critical behaviors. Finally, 19 applications do not deploy SSL/TLS protocols on any critical behaviors. Nine of them use proprietary protocols, Wechat, QQ, QQ music and other applications published by Tencent cooperation included. The rest of ten applications use http protocol on critical behaviors, causing lots of security problems. For example, MengDian, an e-commerce application, exposes users' payment password by plaintext. Table 2 lists applications on iOS platform which leak users' sensitive data.

Table 2. List of applications leaking users' sensitive data on iOS platform.

Application	SSL deployment	Sensitive data exposed	Form (in http packets)
Baiduyun	None-insecurity	File uploaded and downloaded	Ciphertext
Mengdian	None-insecurity	Payment password	Plaintext
Dazhongdianping	None-insecurity	User name, login password	Ciphertext
Kuwo music	None-insecurity	User name, login password	Ciphertext
Kuaishou	None-insecurity	User name, login password	Ciphertext
XimalayaFM	None-insecurity	Phone number; login password	Plaintext; Ciphertext
Kugou music	None-insecurity	Phone number; login password	Plaintext; Ciphertext
Mojitianqi	None-insecurity	Phone number	Plaintext
Souhu vedio	None-insecurity	Login password	Plaintext
QQ mail	None-insecurity	Some details of email	Ciphertext
Qunaer travle	Part	Login password	Plaintext
Fanli	Part	Login password	Plaintext
Mogujie	Part	Login password	Ciphertext

SSL deployment on Android platform has similar status with that on iOS. As shown in Fig. 3, 32 applications deploy SSL/TLS protocols on all critical behaviors. Six applications only deploy SSL/TLS on part of critical behaviors and 12 applications do not deploy SSL/TLS on any critical behavior, of which nine use proprietary protocols and three use http protocol. Table 3 lists applications leaking users' sensitive data on Android platform in detail.

In addition, applications with navigation functions such as Baidu map leak users' accurate location by plaintext in http packets. However, considering the large cost of deploying SSL/TLS on navigation, we do not define navigation as critical behavior. But these applications still need to make improvements on protecting users' location privacy.

From the results, we can see that there are a large number of applications that expose users' sensitive data. Specially, entertainment applications account for a substantial amount. E-commerce and tool applications with security issues also account for

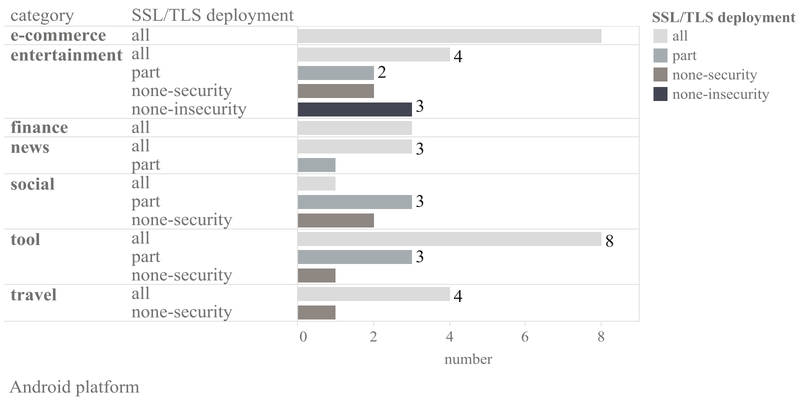

Fig. 3. SSL deployment of different categories on Android platform

Table 3. List of applications leaking users' sensitive data on Android platform.

Application	SSL deployment	Sensitive data exposed	Form (in http packets)
Kuwo music	None-insecurity	User account; login password	Plaintext
Souhu video	None-insecurity	Login password	Plaintext
Letv video	None-insecurity	User account, login password	Plaintext
XimalayaFM	Part	Phone number; login password	Plaintext; ciphertext
Baofeng video	Part	Phone number; login password	Plaintext; ciphertext
Baidu map	Part	User name; login password	Plaintext; ciphertext
Baidutieba	Part	User name; login password	Plaintext; ciphertext
Baiduyun	Part	User name; login password	Plaintext; ciphertext
Sina weibo	Part	Some detail of chat content	Plaintext;

a large number. Application designers should be aware of the significance of deploying SSL/TLS protocols on critical behaviors and pay particular attention to those categories with serious security problems.

4.2 SSL Handshake Parameters

On iOS platform, 37.84% applications use TLS 1.0 version. Sina weibo and Qunaer travel completely use TLS 1.0 version and the others use a mix of TLS 1.0, 1.1 and 1.2 version. 5.4% applications use version 1.1 and 56.76% use version 1.2. There are 12 applications using RC4 encryption algorithm, which is weak and should not be used anymore [13]. Among them, Sina weibo and ICBC mobile bank use TLS 1.0 version as well as RC4 algorithm. This combination of parameters has great risks of being compromised. It is especially dangerous for ICBC mobile bank as a finance application. Besides, one application, 12306 railway, uses self-signed certificates.

On Android platform, 18.75% applications use TLS 1.0 version and the rest use TLS 1.2 version. There are seven applications using RC4 algorithm and two applications using self-signed certificates, which are 12306 railway and Iqiyi.

Table 4 shows the selection of SSL handshake parameters on iOS and Android. Due to many applications use not only one type of encryption algorithm, the sum of three algorithms in Table 4 is more than 100%.

Table 4. Selection of SSL handshake parameters on iOS and Android.

Platform	TLS1.0	TLS1.1	TLS1.2	AES_GCM	AES_CBC	RC4
iOS	37.84%	5.4%	56.76%	75.68%	27.03%	32.43%
Android	18.75%	\	81.25%	71.05%	34.21%	18.42%

The result shows that most applications prefer to use TLS 1.2 version and AES_GCM algorithm, which is a good phenomenon for application security. Thankfully, there is no application using SSL 2.0 or 3.0 version on neither of platforms. However, there are still many applications using the insecure algorithm, RC4. A combination of low TLS version and insecure algorithm can greatly reduce application security. The designers should attach importance to this.

4.3 MITM Experiment Results

In Sect. 4.2 we find that two applications use self-signed certificates, 12306 railway and Iqiyi. We perform MITM experiments on them, as well as four widely used applications from e-commerce and finance categories.

First, 12306 railway and Iqiyi can be compromised. When MITM attack is performed, there is no warning of abnormality and we can use the applications as usual. SSLsplit intercepts their SSL traffic and decrypts it. We can see sensitive data by ciphertext in decrypted packets.

Secondly, two e-commerce applications, Taobao and Jingdong cannot be compromised. SSLsplit hasn't intercepted their SSL traffic and they can work normally. We owe it to their use of proprietary protocols. However, there are still serious problems in Taobao. We can login a user's Taobao account using cookies in http packets instead of the user's account and password.

Finally, the finance application, Jingdong finance, can be compromised as well. But another finance application, Zhifubao, can resist MITM attack successfully. It warns the user with a certificate problem notification and prevents users to use it unless there is no MITM attack any more.

Therefore, the use of self-signed certificates is very dangerous and may be easily compromised by MITM attacks. Of course a successful resistance to MITM attack cannot guarantee communication security completely. However, the designers should avoid using self-signed certificates in applications.

4.4 Discussion

With the measurement results in Sect. 4.1, we try to compare the differences of application security between iOS and Android. However, we find that such a comparison is irrelevant. The first reason is that the popular top 50 applications are different on two platforms. The second reason is that we cannot guarantee an application has the same version on two top application lists. Without variable controlled, the comparison is meaningless.

Moreover, the measurement results in Sect. 4.2 are not complete. There are other correlations between SSL handshake parameters and communication security that we haven't taken into account.

Therefore, in the future, we plan to perform a variable controlled experiment to explore the differences of mobile security in terms of SSL/TLS among different platforms. Besides, we will improve our measurement on SSL handshake parameters and certificate validation of mobile applications.

5 Conclusion

In this paper, we measure the SSL/TLS deployment of the popular top 50 applications on iOS and Android platform. A system is designed and implemented to detect whether an application deploys SSL/TLS protocol and extract the SSL handshake parameters. The results show that 28% of the applications on iOS platform and 20% on Android platform have problems in deploying SSL/TLS protocols on critical behaviors, which can cause severe sensitive data exposure. By analyzing the SSL handshake parameters, we find that most applications use a high TLS version, but some still use weak algorithms and insecure certificates. Then, we perform MITM experiments against applications deploying insecure certificates and several representative applications. The results prove our guess that the former can be easily compromised by MITM attack. We propose concrete recommendations to designers based on our measurement results. Finally, we summarize the shortcomings of our measurement and describe the future work.

Acknowledgments. This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06030200), Xinjiang Uygur Autonomous Region Science and Technology Project (No. 201230123), and Beijing Natural Science Foundation (4164089).

References

1. CNNIC 37th Statistical Report on Chinese Internet. http://tech.sina.com.cn/z/CNNIC37/
2. Ren, J., Rao, A., Lindorfer, M., et al.: Recon: revealing and controlling privacy leaks in mobile network traffic. arXiv preprint arXiv:1507.00255 (2015)
3. Trummer, T., Dalvi, T.: Mobile SSL failures (2015)
4. He, B., Rastogi, V., Cao, Y., et al.: Vetting SSL usage in applications with SSLINT. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 519–534. IEEE (2015)

5. Sounthiraraj, D., Sahs, J., Greenwood, G., et al.: SMV-hunter: large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in android apps. In: Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS 2014 (2014)
6. Balebako, R., Jung, J., Lu, W., et al.: Little brothers watching you: raising awareness of data leaks on smartphones. In: Proceedings of the Ninth Symposium on Usable Privacy and Security, p. 12. ACM (2013)
7. Roesner, F., Kohno, T., Wetherall, D.: Detecting and defending against third-party tracking on the web. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, p. 12. USENIX Association (2012)
8. Egele, M., Kruegel, C., Kirda, E., et al.: PiOS: detecting privacy leaks in iOS applications. In: NDSS Network and Distributed System Security Symposium (2011)
9. Levillain, O., Ébalard, A., Morin, B., et al.: One year of SSL internet measurement. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 11–20. ACM (2012)
10. Pukkawanna, S., Kadobayashi, Y., Blanc, G., et al.: Classification of SSL servers based on their SSL handshake for automated security assessment (2014)
11. Georgiev, M., Iyengar, S., Jana, S. et al.: The most dangerous code in the world: validating SSL certificates in non-browser software. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 38–49. ACM (2012)
12. Transparent SSL/TLS interception. http://www.roe.ch/SSLsplit
13. Sheffer, Y., Holz, R., Saint-Andre, P.: Summarizing known attacks on transport layer security (TLS) and datagram TLS (DTLS) (2015)