J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

Method to Analyze Information Leakage Malware using SSL Communication in Android Platform

Gilsu Cho[1] Sangwho Kim[1] Jaecheol Ryou[1*]

ABSTRACT

Widely used around the world, smartphones contain many features and can store content such as contacts, photos, and videos. Information that can be leaked in proportion to the information that the smartphone can store has also been increased. In recent years, accidents such as personal information leakage have occurred frequently. Personal information leakage is happening in the Android environment, which accounts for more than half of the smartphone operating system market share. Analyzing malicious apps that leak information can tell you how to prevent information leakage. Malicious apps that leak information will send importantinformation to the hacker's (C & C) server, which will use network communication. Malicious apps that are emerging nowadays encrypt and transmit important information through SSL communication. In this case, it is difficult to knowwhat kind of information is exposed to network. Therefore, we suggest a method to analyze malicious apps when leak important information through SSL communication. In this paper, we identify the way malicious apps leak information. And we propose a method for analyzing information leaked by SSL communication. Data before encryption was confirmed in the device through SSL hooking and SSL Strip method.

☞ keyword : Android, Information leakage, network packet monitoring, SSL, malware

1. Introduction

Widely used throughout the world, smartphones are portable and lightweight, offering basic features such as phone and messaging, web surfing, and camera. Inside the smart phone, various kinds of data related to these services are stored. Especially, important information such as schedule and account information is stored in the smartphone, and the damage is great when information is leaked.

Analyzing malicious apps that leak information can tell you how to prevent information leakage. There are static and dynamic methods for analyzing malicious apps. The static analysis method is to analyze the source code of the malicious application, and the dynamic analysis refers to the method of checking the memory or confirming the packet while driving the malicious application in the device. Malicious apps that leak information will send important information to the server (C & C) of the hacker, which will use the network communication. In other words, when malicious apps are running, you can better understand what information is being leaked by looking at network communication. However, there is a case where malicious apps appearing recently are encrypted by transmitting important information through SSL communication. In this case, it is difficult to know what kind of information is exposed by network communication. Therefore, we suggest a method to analyze malicious apps when malicious apps leak important information through SSL communication.

The composition of this paper is as follows. Chapter 2 introduces related research, and Chapter 3 analyzes information leak pattern of Android malicious apps. Chapter 4 proposes a method for analyzing malicious apps when they communicate with SSL, and evaluates the method presented in Chapter 5. Finally, we conclude in Chapter 6.

1 Computer Science and Engineering, Chungnam National University, Daejon, 34134, Korea
* Corresponding author (jcryou@cnu.ac.kr)
[Received 23 October 2017, Reviewed 6 November 2017(R2 5 February 2018), Accepted 17 April 2018]
☆ This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2014-6-00909,Research on the security of operating system).
☆ A preliminary version of this paper was presented at APIC-IST 2017 and was selected as an outstanding paper.

2. Related work

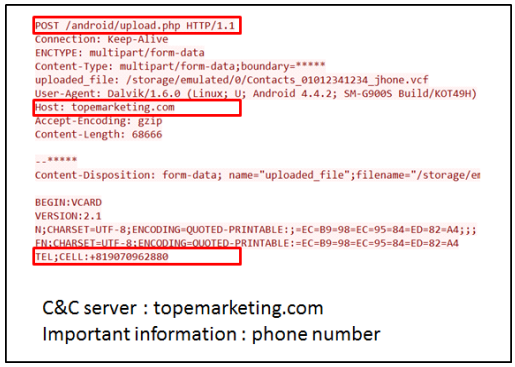
There is article in the Android that want to detect malicious apps through the network[1]. In this paper, make important information such as imei on network packets through regular expressions. And filter the data to be transmitted in GET or POST method innetwork communication. This method can detect information leaks that are not encrypted or encoded. However, if a malicious application leaks information in the same way as encoding or encryption, the method presented in the paper can not detect information leakage.

3. Malicious app information leak pattern

When analyzing malicious apps that leak important information into the device, it canbe confirmed that there are three patterns of leakage of important information. The first is information leakage through plain text communication, the second is information leakage through encoding, and the third is information leakage through encrypted communication using SSL. In each case, we identified how to analyze malicious apps.

3.1 Information leakage through plaintext

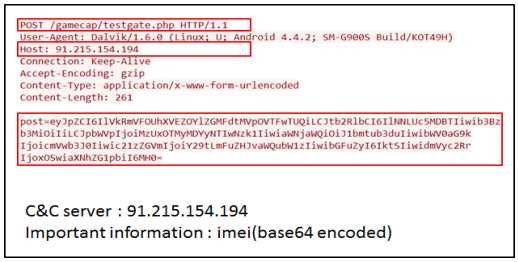
(Figure 1) is a packet when a malicious app runs. When analyzing the packet, we can confirm that the contact information is transmitted through the http plaintext communication. In this case, you can check the C & C server and important information.



(Figure 1) Communication through plain text

3.2 Information leakage through encoding

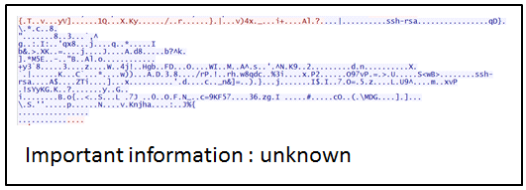
Malicious apps may also encode important information when it is leaked (Figure 2). When the encoded data is decoded, the imei value of the corresponding device can be confirmed. In this case, unlike plain text communication, it is difficult to immediately check which information has been leaked. However, if it is known how to encode it, it is easy to obtain data through decoding.



(Figure 2) Communication through encoding

3.3 Information leakage through encryption

(Figure 3) is a packet when a malicious app that transmits important information encrypted with SSL is activated. In this case, since it is encrypted, it is difficult to confirm what kind of information is actually leaked by only the network packet. Therefore, it is necessary to find out how to confirm such encrypted information.



(Figure 3) Communication through encryption

4. Method to analyze malicious apps using SSL

SSL (Secure Socket Layer) is a protocol designed to

prevent eavesdropping, interference and forgery while communicating between client and server network. Therefore, even if you capture a packet over a network communication, you does not know what contents are communicated.Such SSL is widely used in browsers and various applications. The SSL protocol has the advantage of being secure from eavesdropping, interfering, and counterfeiting. However, if a hacker exploits this point and uses SSL protocol to leak important information to the network, there is a problem that is difficult to detect in the firewall or the control system through the network.

In fact, according to an analysis of malicious apps using SSL communication in trendmicro in 2014[2], the certificate that malicious apps use to use SSL communication is not a private certificate, but use public services include email, the cloud, and social networks. Because these public services are basically SSL communication, malicious apps use a method of transmitting important information to these services. ANDROID_GMUSE.HNT, one of the malicious apps that useGoogle gmail service. It hard-code the gmail account and password in the app, and send the important information encrypted by SSL communication to the hacker's Google mail as shown in (Figure 4).

If the important information is transmitted in this manner, it is highly likely that the packet is analyzed on the network and it is confirmed as a normal mail transmission, and the content of the data is not known.

```
private Vector filelist = new Vector();
private String filename = "";
private TelephonyManager mTelephonyManager;
private String mailhost = "smtp.gmail.com";
private String password = [redacted];
private Session session;
private String user = [redacted]gmail.com";

static
{
  Security.addProvider(new JSSEProvider());
}

public GMailSender()
{
  Properties localProperties = new Properties();
  localProperties.setProperty("mail.transport.protocol", "smtp");
  localProperties.setProperty("mail.host", this.mailhost);
  localProperties.put("mail.smtp.auth", "true");
  localProperties.put("mail.smtp.port", "465");
  localProperties.put("mail.smtp.socketFactory.port", "465");
  localProperties.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
  localProperties.put("mail.smtp.socketFactory.fallback", "false");
  localProperties.setProperty("mail.smtp.quitwait", "false");
  this.session = Session.getDefaultInstance(localProperties, this);
}

protected PasswordAuthentication getPasswordAuthentication()
{
  return new PasswordAuthentication(this.user, this.password);
}
```

(Figure 4) ADNROID_GMUSE.HNT

4.1 SSL hooking

Hooking is one of the methods used to identify data encrypted with SSL. Hooking is a term of software engineering that refers to commands, methods, and techniques that interchange or intercept function calls, messages, and events that occur between software components in various computer programs, such as operating systems or application software. Hooking can be used in an attack in the same way as hook the input value of the keyboard to leak out the user's password or to extract important information from the memory. However, if you use these hooks well, you can figure out what information is being leaked by an application that uses SSL to encrypt information. In order to hook in Android, Android rooting is basically necessary. Android does not provide users with a root privilege, which is basically the highest privilege. It is said that the user arbitrarily changes the device to obtain the root privilege. Hooking requires rooting because it requires privileges that can not be overridden, such as modifying values n memory or modifying files.

In order to hook in Android, I analyzed open source tools that are applicable to older Android devices [4]. Hooking in the toolattaches to a specific process and loads a library into the process memory. Then, when the function to be hooked is called, the function of the library loaded in the memory is called, and then hooking is performed in a form that calls a normal function again. In this way, you can check the plaintext data by hooking the ssl_write function in the gmail process in nexus, 4 4.2 (Figure 5).

```
SSL_wrtie() called buf : GET /mail/gxlu?email=gogogo%40naver.com&zx=1488631598135 HTTP/1.1
Host: mail.google.com
Connection: keep-alive
Referer: https://accounts.google.com/signin/v1/lookup
X-Requested-With: com.explore.web.browser
User-Agent: Mozilla/5.0 (Linux; U; Android 4.3; ko-kr; Nexus 4 Build/JWR66Y) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30
Accept-Encoding: gzip,deflate
Accept-Language: ko-KR, en-US
Accept-Charset: utf-8, iso-8859-1, utf-16, *;q=0.7
Cookie: NID=98=4HYSOGrcLrLBlu-mLYJyCHnNuYZZ0sy6_9QxS9yGmORGdW_Y5TvyHu_zyCDRvpPC_imZQxn8yJgsXOuSR4mFIgp8Nmwy2nRWgd2FRs6tc21azGMJg_xEGZUJzkbDmI1Z4f4naz1B0UU; GMAIL_RTT=471
```

(Figure 5) Hooking, nexsus, 4.4.2

However, there is a disadvantage that the tool can only hook a specific function ina specific process, and since it has an assembly language and is implemented as an arm32 assembly, it is not applicable to arm64 devices. But the

selinux[5] security policy started from Android 4.3 and completely applied to 5.0 does not allow you to excute the library loaded in memory or write a log file using the corresponding tool on the Android 6.0.1 device (Figure 6).

```
: avc:  denied  { write } for  pid=9797 comm="main" name="arm" de
: avc:  denied  { execute } for  pid=8986 comm="ogle.android.gm"
: avc:  denied  { execute } for  pid=8986 comm="ogle.android.gm"
```

(Figure 6) actions denied by selinux

Due to the selinux security policy, it is hard to use hooking by load and run the library in memory. One of the other ways to hook in Android is to add a function to output the log to SSL_write function of libssl.so, a library that controls SSL communication. If you use this method, you are not restricted by selinux because you have added log output function which is allowed in untrusted_app which is the selinux context that user app has. And you do not have to worry about architectures like arm32 and arm64 because you only need to get libssl.so file after compiling the source code of the desired device.

Infact, I have been working on replacing the libssl.so file on my Android nexus 6p, 6.0.1 device. To replace the libssl.so file, you need device rooting as mentioned above. After rooting, you have to download the asop firmware for the test device. For Nexus models, the firmware is available on the Android homepage. For the test model, download firmware with the name of android_6.0.1_r62. After downloading the firmware, you need to find the source code for the libssl.so library in the firmware. The code for the test device was in the "external/ boringssl/src/ssl" path, and the SSL_write function to be hooked is in ssl_lib.c. After that, log the output of the SSL_write function of the function to the log before the SSL encryption, as shown in (Figure 7), and add the code to send it to the socket. To use the log output function, you need to change Android.mk file located in "external/ borings".

And compile only the libssl.so file with the "make libssl" command, and replace the existing libssl.so on the device with the modified file. Then, as shown in (Figure 8), it is confirmed that the data before encryption is recorded in the log, and it can be confirmed that the data is also transmitted to the private server. We use socket because the previously recorded log is deleted when some amount of data is accumulated.

```
int SSL_write(SSL *s, const void *buf, int num) {

  FILE *fd;
  int pid;
  int len = strlen(buf);
  int sock;

  struct sockaddr_in server_addr;

  if (s->handshake_func == 0) {
    OPENSSL_PUT_ERROR(SSL, SSL_write, SSL_R_UNINITIALIZED);
    return -1;
  }

  if (s->shutdown & SSL_SENT_SHUTDOWN) {
    s->rwstate = SSL_NOTHING;
    OPENSSL_PUT_ERROR(SSL, SSL_write, SSL_R_PROTOCOL_IS_SHUTDOWN);
    return -1;
  }

  sock = socket(PF_INET, SOCK_DGRAM, 0);

  memset(&server_addr, 0, sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(4747);
  server_addr.sin_addr.s_addr = inet_addr("168.188.123.213");

  LOGD("%s", buf);
  sendto(sock, buf, len, 0, (struct sockaddr*)&server_addr, sizeof(server_addr));

  ERR_clear_system_error();
  return s->method->ssl_write_app_data(s, buf, num);
}
```

(Figure 7) modified source code

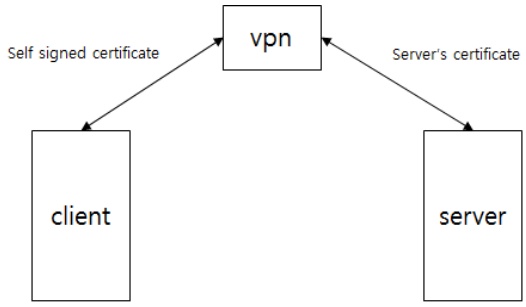
```
X-SERVER-TOKEN: [redacted]
Cookie: NID=114=xHSDAr1xVVZO6Ulync6H2r4toGpT5nT6QmAFgDFZZeMKAIxdmiTKd7QF4EgnPl
User-Agent: Android/com.google.android.gms/11518440 (angler MTC20F); gzip
content-type: application/x-gzip
content-encoding: gzip
Transfer-Encoding: chunked
Host: play.googleapis.com
Connection: Keep-Alive
Accept-Encoding: gzip

2000
KgBOovXrNeAI7J1PHTGBuBNTfWEcNPjjBk8JgvDCHKdrNMGhVbfbf2hP43DvDUDzV1Ant3o8jSg5n:
User-Agent: GmsCore/11518440 (angler MTC20F); gzip
content-length: 592
Host: www.googleapis.com
Connection: Keep-Alive
```

(Figure 8) Hooking, nexsus, 6.0.1

4.2 SSL Strip

SSL Strip method exists as another method of verifying data encrypted with SSL. Unlike hooking, this method has the advantage that it does not need root privilege. This method is basically based on the Man In The Middle (MITM) attack. To do the SSL Strip in the Android device, use the method shown in (Figure 9). Install a virtual private network (VPN) inside Android deviceand use it as a proxy server, and configure the server and client communication through a proxy server. Then, after installing a private certificate on the device, communication between the server and the proxy server takes place with the server's certificate, and communication between the proxy server and the client proceeds with the private certificate.

There is an app called "Packet Capture" exists in the Android Play Store. The app has implemented the same method as above, and the user can use it by installing the certificate and vpn. With this app (Figure 10), you can check the SSL communication of a specific process in plain text form in real time. You can also save the file as a pcap file. Using this tool, you can see what data is dynamically flowing when you analyze malicious apps.



(Figure 9) SSL strip using VPN

(Figure 10) SSL Strip, nexus, 6.0.1

5. Evaluation

We can monitor the communication with SSL through the two methods presented in this paper. SSL hooking has the disadvantage of requiring rooting, but it can log all SSL communications used by the device. SSL Strips have the advantage that no rooting is required, but there is a disadvantage that only SSL communications used by certain processes can be logged. We tried to test our method to malicious app. But we cannot find malicous app file that use SSL communication,if one app use SSL communication, the certificate of public accounts such as gmail or cloud service, the certificate is expired. So, we made and tested arbitrary malicious apps. As a result of testing using two methods presented in this paper for malicious apps that leak imei information through SSL communication, we could confirm the data before encryption.

6. Conclusion

In this paper, we identify the way malicious apps leak information and summarize the methods that can be analyzed when information is leaked by encrypted communication. There are three types of malicious apps that leak information. In the case of plain text communication and encoding communication, it is also possible to identify what information is leaked by capturing packets on the network. However, there is a malicious app that encrypts information by SSL and leak information. In the case of such a malicious app, there is a disadvantage that it is difficult to know what kind of information is leaked only by packet capture.

The two methods presented in this paper can monitor the SSL communication used by the device. From the viewpoint of network control, it is very important to confirm SSL communication, so the method presented in this paper can be utilized. As a future plan, we will develop a module to control malicious app on actual network using the method presented in this paper.

References

[1] Iland D.; Pucher A.; Schauble T. Detecting Android Malware on Network Level. University of California, Santa Barbara, Dec. 2011.

[2] Seven Shen, Android Malware Use SSL for Evasion, 2014, http://blog.trendmicro.com/trendlabs-security-intelligence/android-malware-use-ssl-for-evasion/

[3] Wekipedia, Hooking, 2017, https://ko.wikipedia.org/wiki/hooking

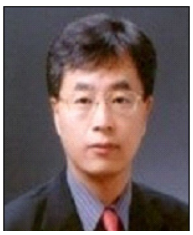
[4] Hijack, adbi – The Android Dynamic Binary Instrumentation Toolkit, 2015,

https://github.com/crmulliner/adbi
[5] Google, Security-Enhanced linux in Android, 2017, https://source.android.com/security/selinux
[6] Shubair Abdulla and Altyeb Altaher, "Intelligent Approach for Android Malware Detection," KSII Transactions on Internet and Information System, vol. 9, no.8, pp. 2964-2983, 2015. https://doi.org/10.3837/tiis.2015.08.012
[7] Yilin Ye, Lifa Wu, Zheng Hong and Kangyu Huang, "A Risk Classification Based Approach for Android Malware Detection," KSII Transactions on Internet and Information Systems, vol. 11, no.2 pp.958-981, 2017. https://doi.org/10.3837/tiis.2017.02.018
[8] Xi Xiao, Zhenlog Wang, Qi Li, Qing Li and Yong Jiang, "ANNs on Co-occurrence Matrices for Mobile Malware Detection," KSII Transactions on Internet and Information Systems, vol. 9, no. 7, pp. 2736-2754, 2015. https://doi.org/10.3837/tiss.2015.07.023
[9] Tae-kyung Ju, Weon Shin. "A New Filtering System against the Disclosure of Sensitive Internal Informaiton" Journal of the Korea Institute of Information and Communication Engineering, 19(5): 1137-1143, May, 2015.

◐ 저 자 소 개 ◑

Gilsu Cho
2017 B.S. in Computer Science and Engineering, Chungnam National University, Korea
2017~Present : M.S. Student in Computer Science and Engineering, Chungnam National University, Korea
Research Interests : System Security, Mobile Security, Vulnerability Analysis
E-mail : happysue0202@gmail.com

Sangwho Kim
2014 B.S. in Computer Science and Engineering, Chungnam National University, Korea
2016 M.S. in Computer Science and Engineering, Chungnam National University, Korea
2016~Present : Ph.D Student in Computer Science and Engineering, Chungnam National University, Korea
Research Interests : System Security, Mobile Security, Vulnerability Analysis
E-mail : whoyas2@cnu.ac.kr



Jae-Cheol Ryou
1985 B.S. in Industrial Engineering, Hanyang University, Seoul, Korea
1988 M.S. in Computer Science, Iowa State University, USA
1990 Ph.D in Computer Science, Northwestern University, USA
1991~Present : Professor, Computer Science and Engineering, ChungnamNational University, Korea
Research Interests : Information Security, Internet Security, Cryptography, Security Protocol
E-mail : jcryou@cnu.ac.kr