Penetration Testing of Transport Layer Security in Android Applications: A Practical Approach

[1] Vijay Kumar Sharma, [2] Dr. Priyanka Sharma

[1] *Student M. Tech (Cyber Security), Department of Information Technology*
[2] *Head of department, IT & Telecommunication, Raksha Shakti University,Ahmedabad,Gujrat,India*

ABSTRACT

In present for shopping, payments, social media, chatting for all work we Android users are mostly use Android apps. As bulk of data is communicated between Android Apps and servers so security of data is a very big concern. The research indicates that the lake of awareness to using transport layer security (TLS/SSL) (encryption) by developers which is leads the data to man-in-the-middle attacks. The data in motion can be captured by a third person by using man in the middle attack so proper tuning of the encryption (TLS/SSL) and keys are very important, In this paper we discussed about the how can we check the flaws of security in Android Apps and this paper focused on the how to secure the Android apps and server communication and what are the main vulnerabilities are occurring in present days though the transport layer encryption is vulnerable. This paper discussed about the main reasons to transport layer vulnerabilities and practical implementation and limitations as well i.e. improper development of android app, Insecure server configuration, unawareness of TLS by the users, the paper discuss the open source tools and lights on usefulness of that tools for testing the TLS flaws In the Android Apps. At the end of the paper we discussed what are the key distance between theory and practical limitations to apply the (TLS/SSL) encryption while data in motion. With the discussions of the TLS flaws paper also give some proper solution regarding that flaw and also discussed about the present research work and the future scope of the research on the topic.

Keyword: - *Android, Android Apps, Servers, Data in motion, Transport Layer Security, Secure Socket Layer, Encryption, Man In The Middle Attack, Vulnerabilities, Configuration, Open source tools, practical*

1. INTRODUCTION

Smartphone is the new key aspect of innovation now in future time the smart phones are all over replace the laptops and personal computers. In the world of smartphone the Android is rises the most popular and powerful operating system among all the reason behind this that the flexibility and power of the Linux and most important that it is open source so everyone can easily afford this .The power and affordability of microelectronics and Android make an incredible combo ,that's why the android smartphone's are in reach of every common person not in India in whole world .But this is now become the area of concern because all of the above specified reason The Android phone 's security is on stake the reasons are many but in the paper we are concern about the security of transport layer security in Android applications. Mostly all of Android users are do all the activities by the using of their Android

apps so the apps are communicate the server and the server reply them just like the web apps.

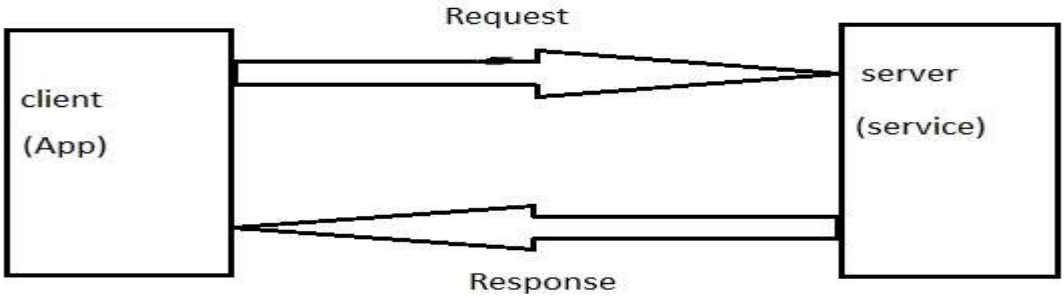


Fig.1 App server Communication

There are various time it comes into news that the Android apps are getting victim of the man in the middle attack here mostly the reason behind this are money, revenge or staking. That's why mostly the payment apps are victimize by man in the middle attack.

1.1 Transport Layer

The transport layer is the fourth layer in OSI (open system interconnection) model which is mainly responsible for the end to end communication over a network. It provides error correction and logical communication between hosts with the help of properly defined protocols.

1.2 Insufficient Transport Layer Protection

We divide the scenario in two sides first is the server side scenario and second is device side scenario on both sides there are following reasons to vulnerable

Server side scenario

1. Using expiry SSL certificate
2. SSL certificate from untrusted third party CA
3. Not using the RC4 and CBC based ciphers
4. Using older version of TLS and SSL
5. Using less complex algorithm for encryption

Device side scenario

1. Exposed framework
2. Using device with root mode
3. Download the application from the untrusted sources
4. Using open /insecure Wi-Fi connections

2. MAN IN THE MIDDLE ATTACK IN REFERENCE TO ANDROID APPS

As technology and the uses of mobile are increased the personal and confidential data flow of information is also increased in now days. These information packets can be sniffed very easily and manipulated when sent in plain text

HTTP over the communication channel. Web browsers are generally are capable to established secure HTTPS connections because they are designed and programed in a manner but the Android apps does not have these type of capabilities. The very wide and popular unguided programing of Android Apps has leave loopholes in their program which use HTTPS calls, According to a survey of Bureau of Labor Statistics US needs 30% more software engineers in future years by 2022.The Android provides some good encryption like large java encryption library as well as third party implementation like Bouncy castle and OpenSSL. In many cases the Android application does not implement the transport layer security or alters the HTTPS calls

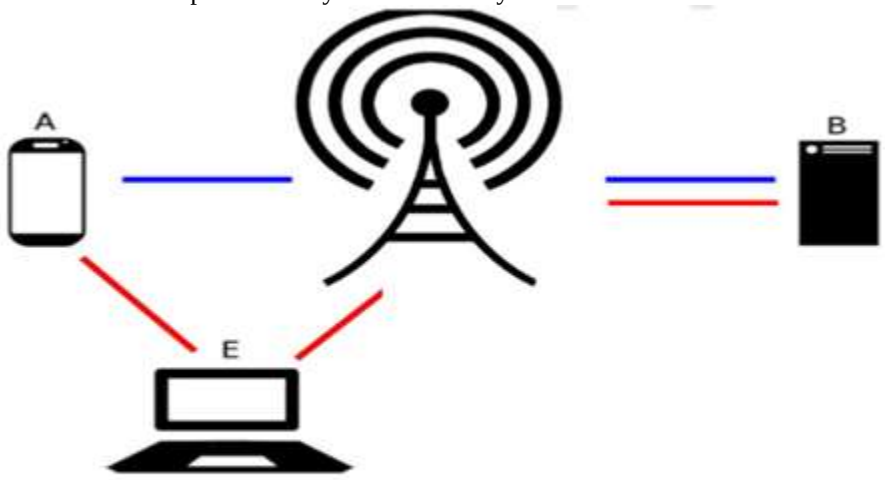


Figure 2 A classic Man In the Middle attack with the conversation between Alice (A) and Bob (B) being intercepted by Eve (E).

Above diagram gives an example of classic man in the middle attack. MITM attack allows third person (E) to sniff, intercept and insert himself into a conversation between two legitimate users (A and B). This is becoming the threat to information security. More in this there is no warning that these vulnerable connections are not secured by SSL/TLS. Issues remains in libraries of the SSL, the X509 is called the certificate validation protocol.

2.1 Android HTTPS and Current Findings

The cryptography is very difficult to implementation in practical manner. In order to create resistant keys, needs complex algorithms and programing methods. Many algorithms are used in different steps in process of encryption. In order to secure information of users as well as integrity developers must be able to use and implement encryption technologies. To make complex the encryption and implement this state of trust a complicated as well as a mixed public and private key exchange takes place. This process required handshake as well as verification process to securely sending the encryption key to any interceptor on the channel who have latched on to the chain of communication. SSL works as follows that first client sends an HTTPS request to the server with its SSL version and supported ciphers and the server is responds with its SSL versions and supported ciphers with a trusted certificate. This server certificate requires a valid sign by a trusted certificate authority (CA) which has verified the serves authenticity. The client will compare the public key of the certificate with its stored local key and the field values to expected values. If certificate passes and the certificate has not been revoked by a CA, the handshake continues. The cipher suite is chosen from algorithms which is the client server have in common. For an example cipher suite can be use ECDHE algorithm for key exchange, RSA algorithm for certificates, and AES128-GCM for

message encryption and SHA256 for message integrity checking.

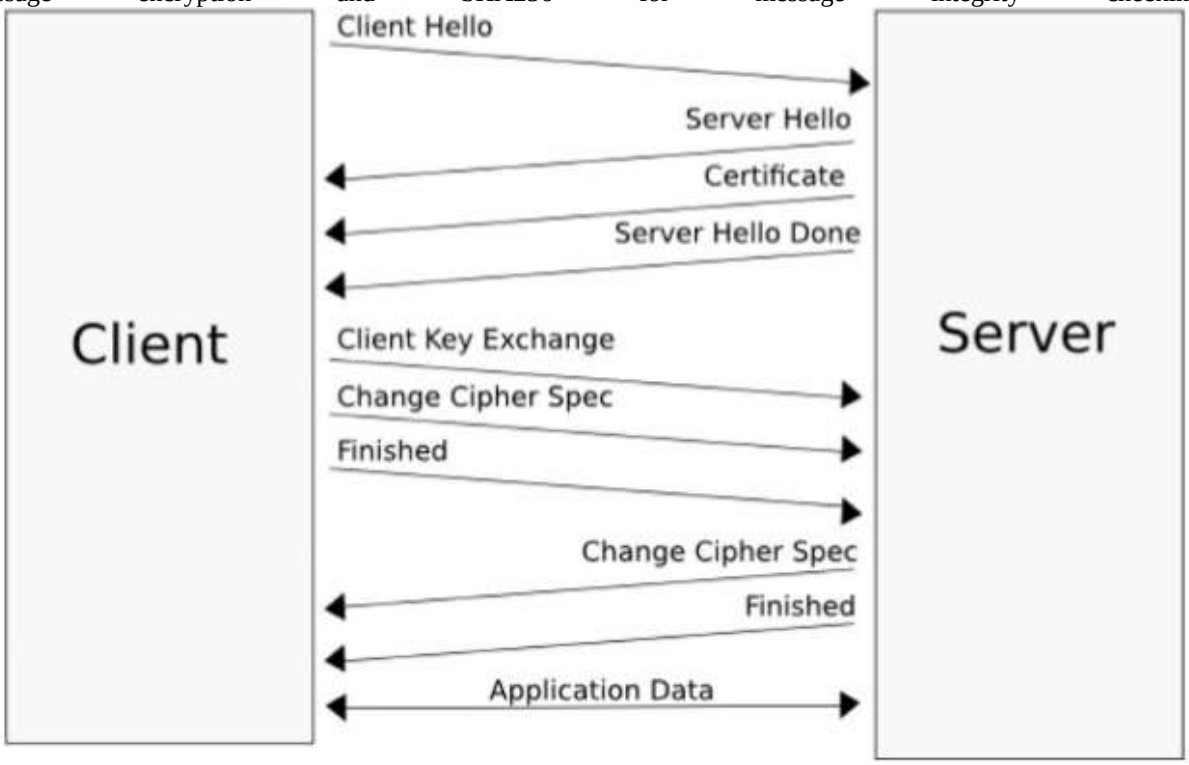


Figure 3 A simple TLS handshake.

If client asked to verify itself with the help of certificate then it add the secret key and transmitted to server over network. If the authenticity is confirmed each machine uses the pre master key to generate the master key for a session ID that functions as the symmetric key for the SSL communication. Once the hand shake has been completed and each device informs that all the communication over the network is will now encrypted with the session id, the client encrypts its message using the symmetric key and sends the data to the server. After the all data is sent over the channel the connection is closed. This can be done through Http Parameters and client connection manager to transmit the proper headers and data. If we selects the cipher suite manually then defaults will be called automatically. A vulnerability note released by CERT identifies Android applications by the Mr. Fandango which fails to find vulnerability in SSL certificate. Fahl et al. developed a tool called MalloDroid which tis useful to analysis application vulnerabilities regarding with MITM attacks. MalloDroid analyzed the API calls whose applications made and checked certificate's validity and also check identity cases of custom HTTPS

implementation.

```
URL url = new URL("https://wikipedia.org");
URLConnection urlConnection = url.openConnection();
InputStream in = urlConnection.getInputStream();
copyInputStreamToOutputStream(in, System.out);
```

Figure 4 Example of a standard Android HTTPS call.

Custom code did not require anything rather than defaults. In mostly all cases, adding the single character would have allowed the application security via using the HTTPS protocol. Firstly noticeable point is regarding at fault to trust manager which are place a loophole to accept all certificates as well as trust all hostnames and ignore SSL errors. Trust managers exist to validate certificates. In this case if the certificate checking is turned off then the security is compromised regarding to MIMA. When a user uses a user defined trust manager than the vulnerability of accept all self-signed certificates has shown to be an issue in the Android community.

3. CAUSES ANALYSIS OF HTTPS / TLS VULNERBALITIES

In this section the paper will shows issues which compromise the security regarding SSL/TLS (HTTPS) implementations in particularly on Android-based device. First we will see at the primary causes of SSL insecurities with current Android HTTPS implementations. The first step in patching these flaws is determining their origin. The causes are exists in the mobile app development, server misconfiguration, Android documentation, SSL/TLS libraries, the SSL/TLS protocol and application awareness to consumers. Extensive research processing for determine the basic cause of each of those factors. This section will investigate each of these causes further.

4. PENETRATION TESTING

Penetration testing is a process or practice to find the vulnerabilities and loopholes of the computer systems or network systems which can be exploited by the attacker or hacker. Penetration testing in Android Applications can be divided into following scenarios

Transport Layer Security	Insecure Transport Layer Protocols
	TLS Authenticity Flaws
	TLS Weak Encryption
	Bypassing TLS Certificate Pinning
	TLS Known Issues - CRIME, BREACH, BEAST, Lucky13, RC4, etc...
	Disable certificate validation

Figure (a) Transport Layer vulnerabilities

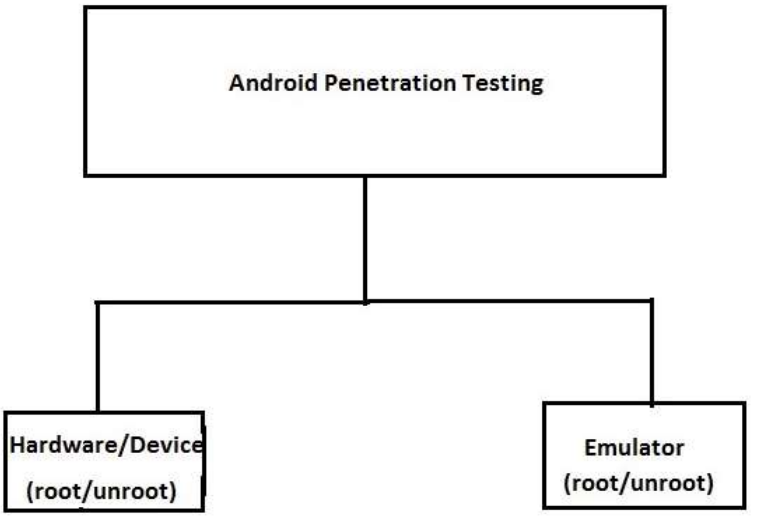


Figure (b) Methods Of Android Petration Testing

For testing the vulnerabilities in any Android App these are the ways which is explained in above Fig but in our paper we are focusing on the Transport Layer vulnerabilities and related issues so first of all we need a lab environment setup for testing the apps so these are the requirements

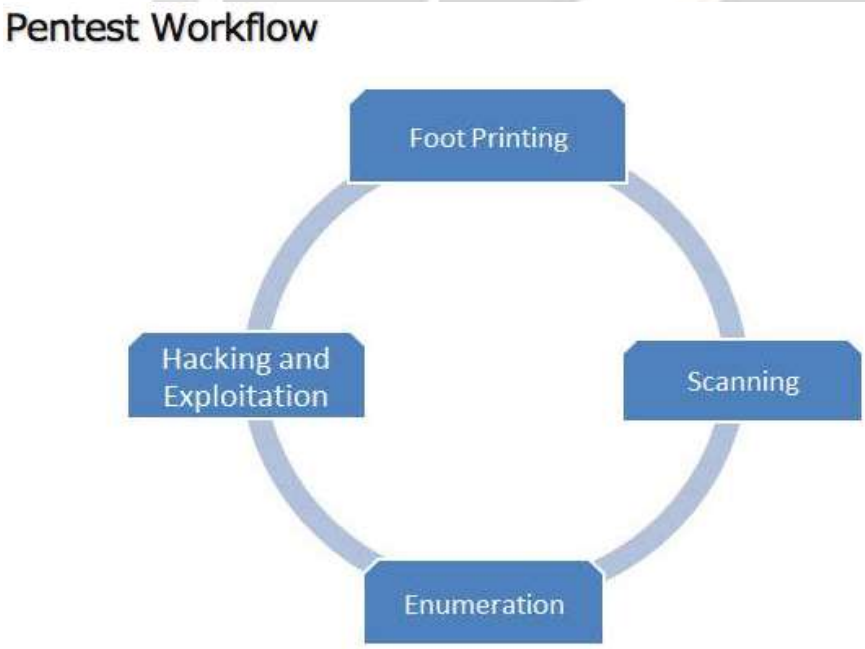


Figure (c) Penetration testing steps

1. Operating system (windows /Linux)

2. Android studio

3. Android SDK Manager

4. Burp suite

5 .Fiddler

6. Wireshark

7. Classyshark

8. Android Debug Bridge

9. Drozer

There are following scenario can be faced during the penetration testing

1. The customer can directly provide the APK file

2. Only source code can be compiled and tested

3. Only a link of App provide by the customer this is coming under the black box testing

The penetration tester have to kept in mind following things when test the App whether

1. The app residing on device
2. The data in motion
3. The data in rest
4. Server communicates with the app

We can use different applications for understanding the concepts of security these apps are free and open source provided for testing purpose for students or professionals to understand and learn the security aspects of security of an Android App . In our case we are majorly focused on the testing of the Transport Layer and related issues which are important. One more reason to use these app and not the commercial apps which we have in our play store is that the copyright issues are there so it is better for learning and practicing the better security we can use the open source project which available as follows

1. Goat Droid project by OWASP: This Project can be downloaded by the Following link https://cloud.github.com/downloads/jackMannino/OWASP-GoatDroid-Project/OWASP-GoatDroid-0.9.zip it is have two apps in it

(i) Four Goats: It is a simple location based social networking app in which we can also check in and check out our locations.

(ii)Herd Financial: It is a financial mobile banking in which user can check their balance status and also transfer the money.

2. Sieve: This is a simple password manager app can be downloaded by the user
https://www.mwrinfosecurity.com/system/assets/380/original/sieve.apk

3. Diva (Dam insecure and vulnerable app): This app have developing time vulnerabilities can be downloaded by http://www.payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz

Now we are going to set up the app in our pc's OS: In a folder hack box we are installing the all above apps and installing the by using adb and Genymotion which is a android emulator you can use
any

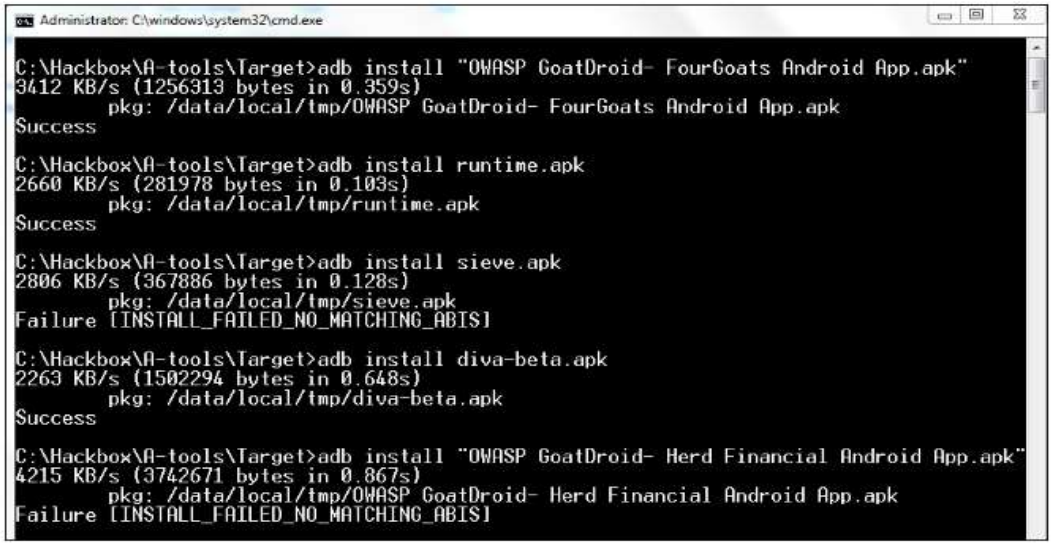


Figure (d)

How to set the backend server: First we have to extract the apk files and after that open with the java command is

Java -jar GoatDroid-0.9.jar

Figure (e)

2. Start it and configure the web services like this

Http port 8888

Https port 9888

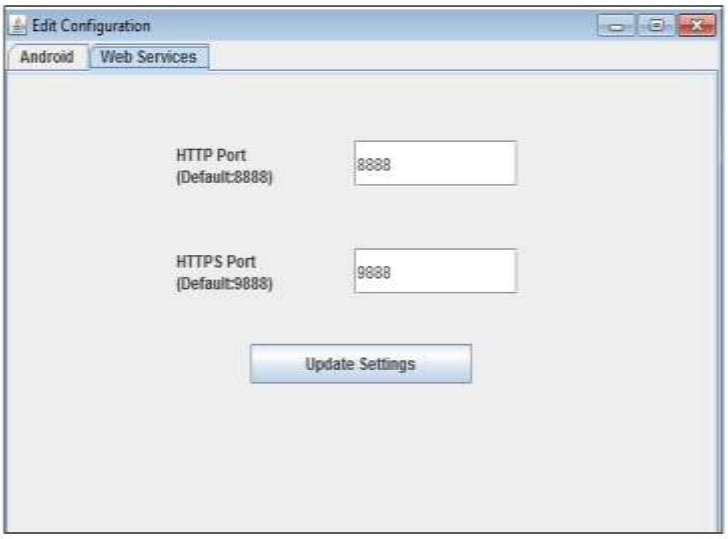


Figure (f)

3. After this we have to set up the application to over emulator genymotion so we have to open the genymotion and type the ip address and port in this

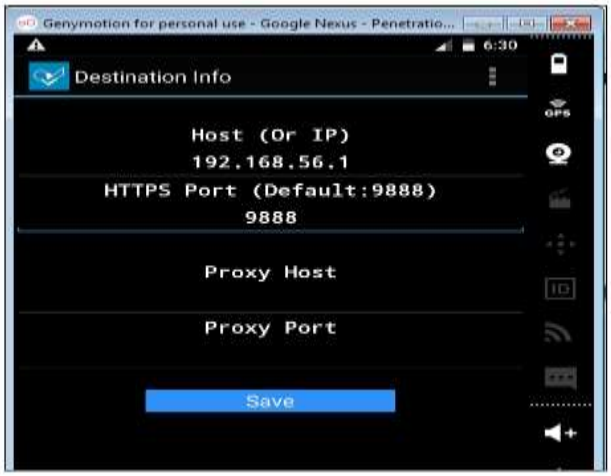


Figure (g)

4. For understanding the apk file in more details we have to disassemble it by this command

Java -jar apktool_2.0.2.jar d “c:/<location of apk file>”

Understanding of the manifest file is very important which is gives the details about the sdk version

After this we have to convert the apk file into jar file by using dext2jar tool the command is following

Dex2jar.bat <name of apk file”

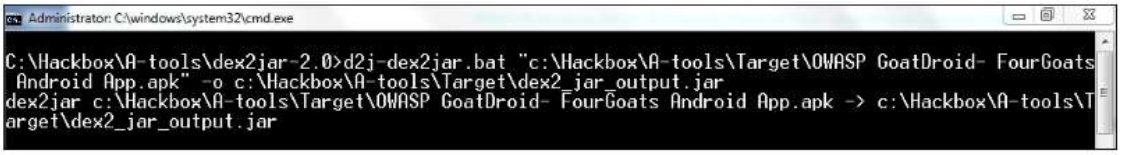


Figure (h)

Now for understanding the souce code of file load this on JD-GUI

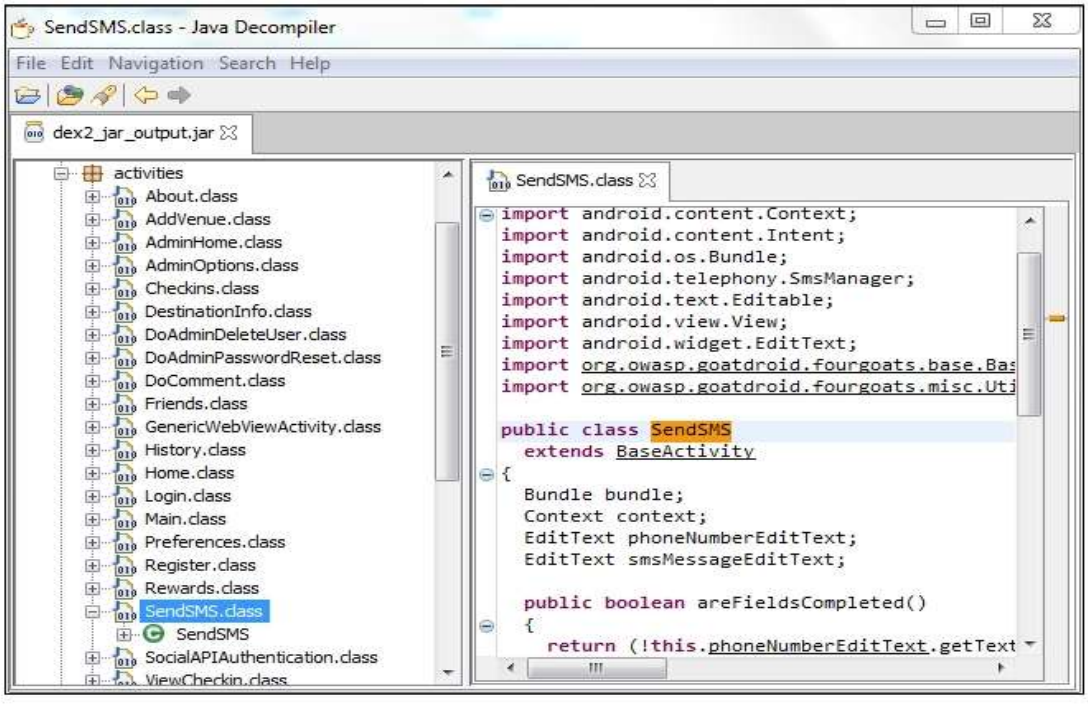


Figure (i)

Understanding of android manifest file can be done by the tool DROZER it have the inbuilt module app.package.manifest gives us the presentable information about the Androidmanifest.xml

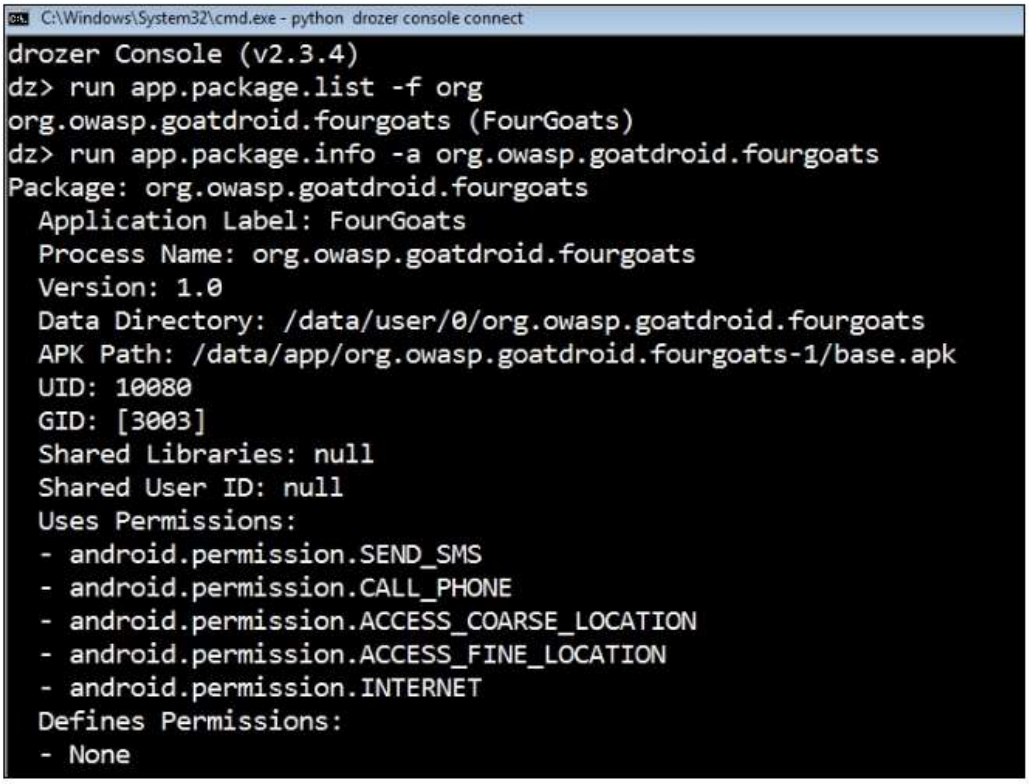


Figure (j)

It contains the information about i.e. Process Name, Data Directory, APK Path, UID and GID, Shared Libraries and Shared User ID

4.1 Transport layer pen testing

1. **MIMA**:

We will setup the proxy and make all the requests and responses go through that particular proxy. Also we will be having an option to manipulate and modify both the packets in the requests and response, and thus assess the application's security

1. In order to create a proxy for HTTP, start the emulator with the -http- proxy flag specify the the proxy ip and port number for example emulator -avd Android_penetesting -http_proxy 127.0.0.1.8080
2. Once we have set up the proxy in the device /emulator, go ahead and launch the Brup proxy in order to intercept the traffic

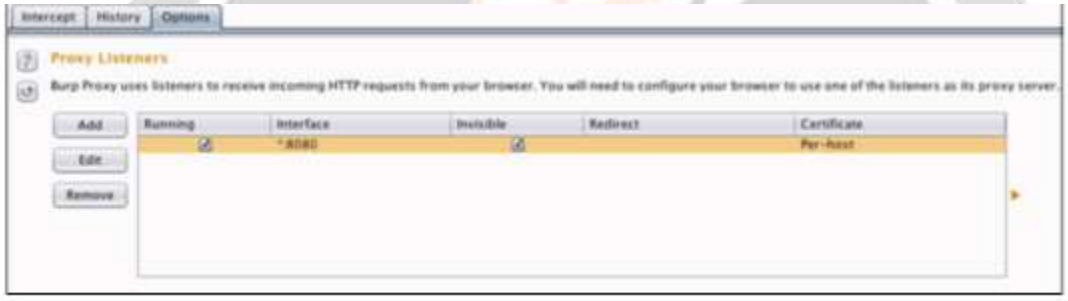


Figure i

In order to check whether the proxy is working or not , open up the browser and launch a website .we will then be able to see if it is getting intercepted or not

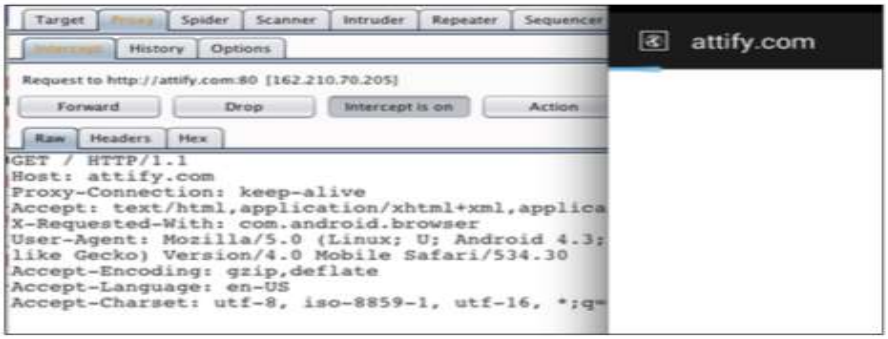


Figure i

HTTPS Proxy Interception

The preceding method will work in the normal traffic interception of application and browser when they are communicating via HTTPS protocol. In https we will get an error due to the certificate mismatch and thus we won't be able to intercept the traffic so for this we have to put a proxy in our Firefox browser

1. To set up proxy in Firefox go to option then advanced tab and then in network

Figure ii

2. Once in the network tab we need to click on settings and configure the settings like this

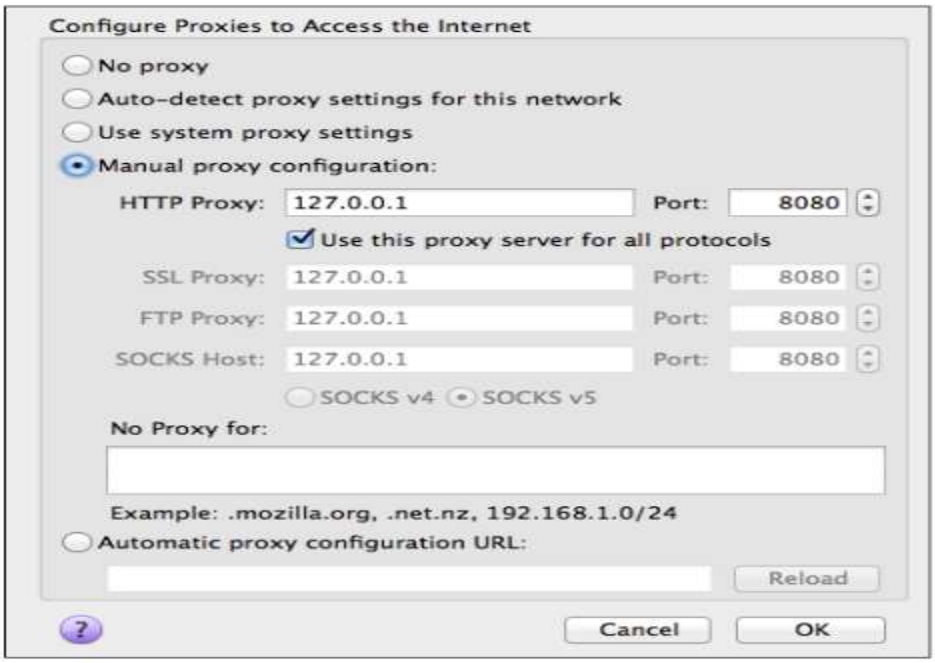


Figure iii

3. Once done go to the HTTPS website on browser on our system here we will receive a network is untrusted message click on i understand the risk and add exception

4. there after click on GET CERTiFICATE and finally ckick on view and then export in order to save the certificate

Figure iv

5. Once the certificate is saved on our system we could now push our device using adb

Adb push portswiggera.crt /mnt/sdcard/portswiggerca.crt

5. Now in our device go to settings and personal categorys there is a option to install the certificate from the sd card install and save the certificate

Figure v

6. Confirm this by going back to our browser and opening an HTTPS website such as https://gmail.com in our case

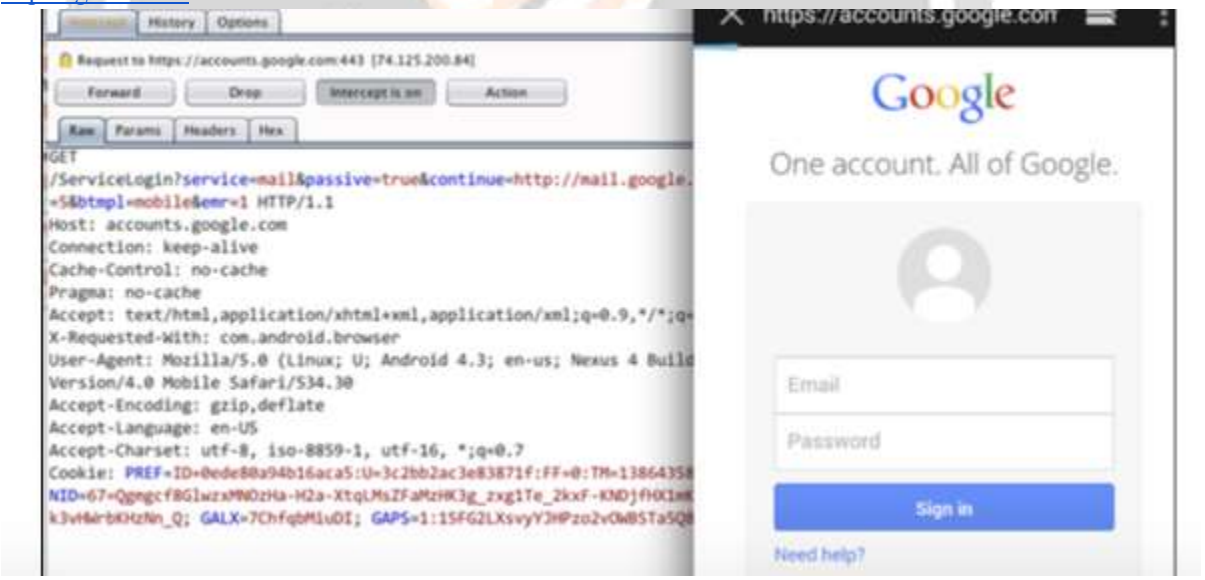
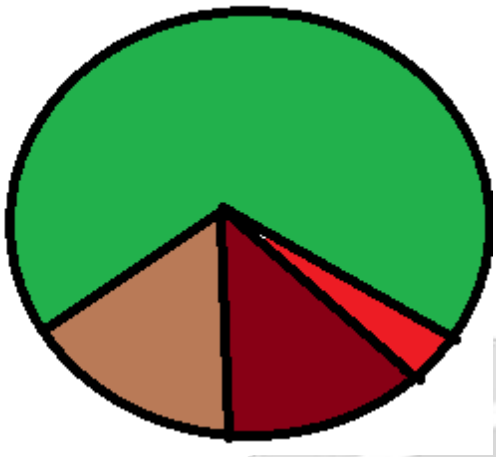


Figure vi

5. RESULT

After Penetration testing 50 apps from the Android play store these are the results as following



Pie chart (Result Analysis)

Green = Secured TLS/SSL Apps (30 Apps)

Brown = Week Encryption (7 Apps)

Purple = Self signed Certificate (8 Apps)

Red = Using HTTP Protocol (5 Apps)

6. CONCLUSIONS

This paper concludes the current workings of TLS/SSL area in reference to Android applications also it provides some areas where the security lacks. Paper provides knowledge that how we can check the flaws of any Android application and improve the security. Paper also shows in result that however we progressed in Transport layer security but somewhere we needs the secure coding and good encryption practice.

6. REFERENCES

[1]. A survey on HTTPS implementation by Android apps: Issues and countermeasures Xuetao Wei, Michael Wolf

[2] Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security **,**SaschaFahl , Marian Harbach, Lars Baumgartner, Bernd Freisleben

[3]. https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

[4]. Penetration testing of android application, Kunjan shah

[5] H. Hwang, G. Jung, K. Sohn, S. Park, A study on mitm (man in 1359 the middle) vulnerability in wireless network using 802.1x and 1360 eap, in: IEEE ICISS, 2008


IJARIIE