ANDROID BASED WS SECURITY AND MVC BASED UI REPRESENTATION OF DATA

Jitendra Ingale,

Parikshit mahalle SKNCOE pune,Maharashtra ,India
Email: jits.ingale@gmail.com

ABSTRACT:

Google's Android is open source; Programmable software framework is subject to typical Smartphone attacks. Such attacks can make the phone partially or fully unusable, cause unwanted changes. While accessing data over web services there should be security mechanisms like encryption of data on server side and decryption using key on client side so that attacks can cause minimal damage to device and data integrity

In the second part we have tried to implement here is that representation of data in UI in MVC architecture so that data can be separated from the representation details and user can view data in a manner whichever gives him/her comfort in analyzing the data.

1. INRTODUCTION

Android is a software stack for mobile devices that includes an operating system, middle-ware and key applications. The Android SDK provides the means and APIs necessary to begin developing applications on the Android platform using the Java programming language. As an operating system for mobile devices and embedded systems, Google's Android—an open source framework—is subject to attacks. These attacks impact users of these sophisticated systems adversely and steal their private information and in some cases damage them. These threats to mentioned sophisticated systems are growing day by day as market for smart-phones is subject to grow over the time. It is estimated that smart-phone viruses can update themselves in less time than time taken by viruses to evolve for traditional computer systems .Thus; the challenge in ensuring smart-phone security is becoming similar to that confronting the traditional computer systems.

So far, limited numbers of users of smart-phone has limited the scale and impact of attacks on smart-phones. But it will increase in years to come. Thus, hackers can gain access to the operating system code.

Again, to represent data in an Android application method employed is very simple and is subject to attacks. But is data is separated from its representation to users then even if representation get manipulated data at server can be safe

DOI : 10.5121/ijcseit.2013.3103

2. MOTIVATION

In an Android application development arena we find that we don't have enough application o work with remote server through encrypted web service module. If we see in Appstore, to maintain UI of an application, there are very few applications that have MVC architecture implemented .In the application I am discussing here I have maintained MVC architecture by separating UI and its representation.MVC approach is an effective way to support multiple presentations of data. Users can interact to each presentation in style that is appropriate to the presentation. The data to be displayed is encapsulated in model object. Each model object may have different view objects associated with it where each view is different display representation of the model. Currently also android OS is in news for security threats that may hamper some business scenarios. I have tried to take care some of those.

3. PROPOSED WORK

Android is an execution environment for simulating mobile devices. Android SDK comes in various levels of security. Various system components in the upper layers use libraries from kernel. Incorporating these libraries in Android applications is achieved via Java native interfaces(JNI). **.**dex (Dalvik-executable) files which are compact and memory-efficient than Java class files. The application framework is written in Java, has Google-provided means as well as extensions or services.

Security mechanisms Incorporated in Android.

Mechanism	Description	Security issue
Linux mechanisms		
POSIX users	Each application is associated with a different user ID (or UID).	Prevents one application from disturbing another
File access	The application's directory is only available to the application.	Prevents one application from accessing another's files
Environmental features		
Memory management unit (MMU)	Each process is running in its own address space.	Prevents privilege escalation, information disclosure, and denial of service
Type safety	Type safety enforces variable content to adhere to a specific format, both in compiling time and runtime.	Prevents buffer overflows and stack smashing
Mobile carrier security features	Smart phones use SIM cards to authenticate and authorize user identity.	Prevents phone call theft
Android-specific mechanisms		
Application permissions	Each application declares which permission it requires at install time.	Limits application abilities to perform malicious behavior
Component encapsulation	Each component in an application (such as an activity or service) has a visibility level that regulates access to it from other applications (for example, binding to a service).	Prevents one application from disturbing another, or accessing private components or APIs
Signing applications	The developer signs application .apk files, and the package manager verifies them.	Matches and verifies that two applications are from the same source
Dalvik virtual machine	Each application runs in its own virtual machine.	Prevents buffer overflows, remote code execution, and stack smashing

Table 1.Security Mechanisms Incorporated In Android [9]

At runtime Android execution environment forms .apk installation file. This is similar to a Java .**jar** file in the way that it holds all code and other resources (like images or manifest) for the application. Android applications are developed in Java based on the APIs the Android software development kit (SDK) provides and there is provision for ensuring correct output in emulator. Further, applications needs to be digitally signed (code and other).enclosed public key successfully verifies the signature.

In general, several security mechanisms are incorporated into the Android framework (see Table 1). We can group them into three different groups: Linux mechanisms, environmental features, and Android-specific mechanisms.

4. PROJECT STATEMENT

Generally, in a large application, where client server architecture has to be employed, we prefer Web Services, and then make XML based RPC calls between the client and the server. XML based RPC calls means that communication string will be in XML format. It is not advised to have tightly coupled server and client.

One way to do this is to create a server that accepts commands and arguments through the use of HTTP POST and returns data as a string. The client can then make calls to the server by RPC and pass data to server.

In this paper we will propose the way to implement simple client/server. As http protocol is not secure one should prefer https i.e. secure http . We have opted to use HTTP for its simplicity and convenience.

The Statement is as follows:

Representation of data in MVC type of UI architecture which separates data representation technique from data collection on client side with information stored on distant server.

Transport layer protocols like TCP and UDP have gained lot of value in today's Internet. TCP i.e. Transmission control protocol provides connection-oriented network, reliable end-to-end communication service. TCP programming in Android uses APIs provided in java.net package.

Creation of TCP server requires a Server Socket instance, and wait for (accept) incoming connections. If there's a connection available, accept method will return open socket.

For generation of client we requires IP and port number. Once connection established, one can receive and send message through the socket.

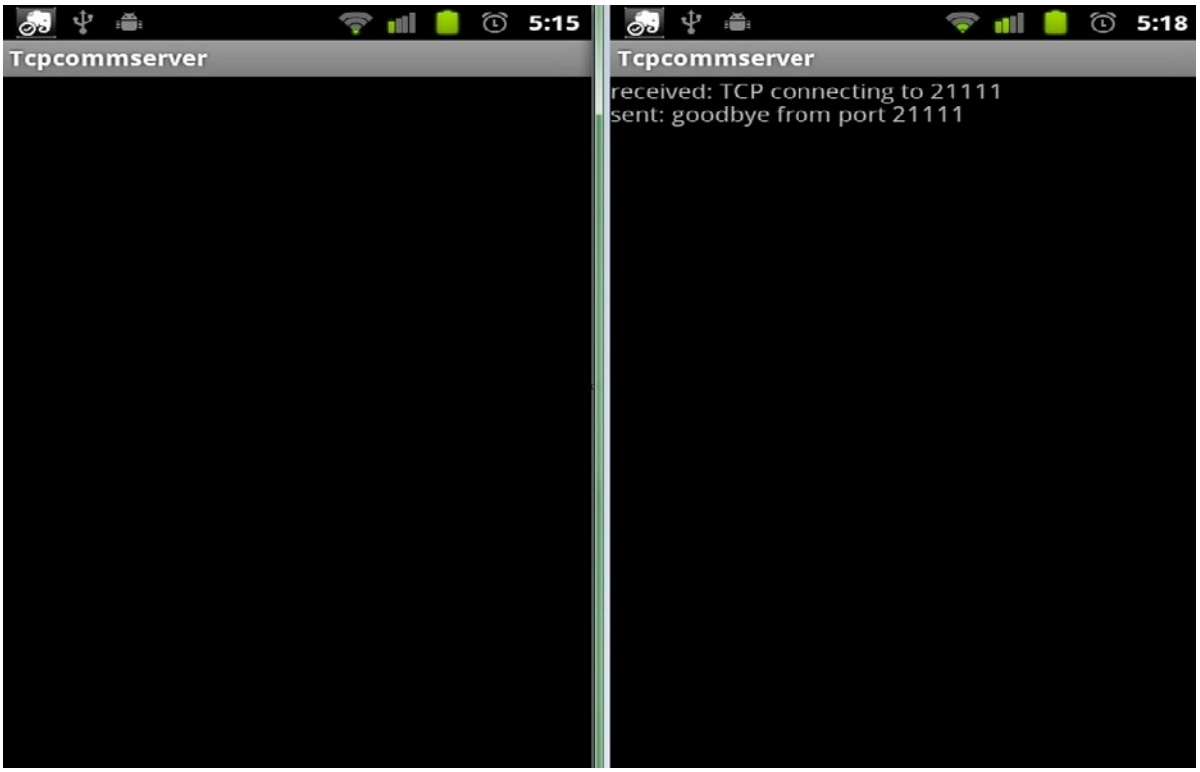

Fig 2 TCP connection

Above diagram shows communication between client and server

5. SOLUTION TO THE PROBLEM

Rather than using the 3rd party API's, json classes etc. we will use default HttpClient from org.apache.http package. The code will be as follows:

```
HttpClient httpclient = new DefaultHttpClient();
HttpGet httpget = new HttpGet("http://example.com/script.php?var1=androidprogramming");
try {
   HttpResponse response = httpclient.execute(httpget);
   if(response != null) {
      String line = "";
      InputStream inputstream = response.getEntity().getContent();
      line = convertStreamToString(inputstream);
      Toast.makeText(this, line, Toast.LENGTH_SHORT).show();
   } else {
      Toast.makeText(this, "Unable to complete your request", Toast.LENGTH_LONG).show();
   }
} catch (ClientProtocolException e) {
    Toast.makeText(this, "Caught ClientProtocolException",
Toast.LENGTH_SHORT).show();
```

```
} catch (IOException e) {
   Toast.makeText(this, "Caught IOException", Toast.LENGTH_SHORT).show();
} catch (Exception e) {
   Toast.makeText(this, "Caught Exception", Toast.LENGTH_SHORT).show();
}
```

Client server communication is easy in android environment.This is done as follows

- Create HttpClient with the default constructor.
- Create a HttpGet or HttpPost object as per requirement, here we have made a GET object so that we can know the status
- Object initialization.
- Execute the GET/POST object through the Http and server will response in the response object of HttpResponse.

Fetching Data From HttpResponse:

To receive data from HttpResponse we should parse it into Inputstream.

```
private String convertStreamToString(InputStream is) {
   String line = "";
   StringBuilder total = new StringBuilder();
   BufferedReader rd = new BufferedReader(new InputStreamReader(is));
   try {
      while ((line = rd.readLine()) != null) {
         total.append(line);
      }
   } catch (Exception e) {
      Toast.makeText(this, "Stream Exception", Toast.LENGTH_SHORT).show();
   }
   return total.toString();
}
```

below given is MD5 hashing in java for encryption of string data.

```
public String MD5(String md5) {
  try {
      java.security.MessageDigest md = java.security.MessageDigest.getInstance("MD5");
      byte[] array = md.digest(md5.getBytes());
      StringBuffer sb = new StringBuffer();
      for (int i = 0; i < array.length; ++i) {
        sb.append(Integer.toHexString((array[i] & 0xFF) | 0x100).substring(1,3));
      }
      return sb.toString();
   } catch (java.security.NoSuchAlgorithmException e) {
   }
      return null;
}
```

6. CONCLUSION

Security mechanisms incorporated in Android aim to tackle a broad range of security threats. To further harden up Android devices and enable them to cope with high-risk threats, we proposed several security countermeasures. We subsequently tried for communication between client and server through encrypted md5 hashing and displayed the data in MVC way. For MVC i.e. model view control we separated data from representation

Android security should provide a way that can avoid damage to Linux-kernel layer. Several vulnerabilities and bugs have already exploited these pathways to gain and maintain root permission on the device. Second, the platform needs better protection for hardening the Android permission mechanism or for detecting misuse of granted permissions. We recommend the remote management, VPN, and login solutions to provide telecom operators with a competitive edge when targeting corporate customers.

7. REFERENCES:

[1] "Google Android: A Comprehensive Security Assessment" Published by IEEE Computer and Reliability Societies
[2] C. Dagon, T. Martin, and T. Starner, "Mobile Phones as Computing Devices: the Viruses Are Coming," IEEE Pervasive Computing, vol. 3, no. 4, 2004, pp.11–15.
[3] J. Cheng et al., "SmartSiren: Virus Detection and Alert for Smartphones," Proc. 5th Int'l Conf. Mobile Systems, Applications and Services (MobiSys 07), ACM Press, 2007, pp. 258–271.
[4] A. Gostev, "Mobile Malware Evolution: An Overview," Viruslist.com, 2006; www.viruslist.com/en/analysis?pubid=200119916.
[5] E.E. Schultz, "Where Have the Worms and Viruses Gone? New Trends in Malware," Computer Fraud and Security, vol. 2006, no. 7, 2006, pp. 4–8.
[6] S.Z. Beguelin,G.Brtarte,and C.luna "A formal specification of MIDP 2.0 security model" in the fourth international Workshop of Formal aspects in security and trust
[7] EA. Shabtai, Y. Fledel, U. Kanonov, et al. "Google Android: A Comprehensive Security Assessment", IEEE Security and Privacy, 8(2):35-44, 2010
[8] X. Zhang, O. Aciiçmez, and J. Seifert. "A Trusted Mobile Phone Reference Architecture via Secure Kernel". In Proceedings of ACM workshop on Scalable Trusted Computing, November 2007.