International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 11 (2014), pp. 1079-1084

http://www. irphouse.com

Survey of Cross-site Scripting Attack in Android Apps

Stanzein Sedol[1] and Rahul Johari[2]

[1, 2]University School of Information and Communication Technology,
Guru Gobind Singh Indraprastha University,
Sector 16 - C, Dwarka, Delhi, INDIA

ABSTRACT

The recent advancement in the field of Smartphone has enabled greater integration with the online services to users. This new environment has also lead the developers to extend the online services to phone seamlessly. The most popular and widely used Smartphone OS are Android and iOS. This openness to access and download different types of online applications in our Smartphone has often put our security at stake. Among different types of attacks being performed on our Smartphone in order to extract our sensitive data and information, the top ranked attacks are the Cross-site scripting (XSS) attack and SQL injection attack as listed in Open Web Application Security Project vulnerability list (OWASP) [1].

In XSS attack, the attacker runs malicious code in the WebView component of victims Smartphone. Therefore, WebView is an essential component in both Android and iOS phones [3]. It enables the application to display the content of online resources on phone. So in this paper, we discuss the XSS attack, analyze their basic causes and focus on potential solutions.

Keywords-Android Security attacks;XSS attack, OWASP ;

1. INTRODUCTION

There has been a significant and tremendous growth in using Smartphone and tablets over the past few years. Google's Android and Apple iOS are two most widely used platforms in the recent time taking more than 50% of the market share. Of these two, use of Android is increasing at a large scale because of its simplicity in using and enabling downloading the apps from Google market. And it is also convenient for the developers to design and add their applications in the app store. More and more people now own a Smartphone or a tablet because of the attractive features these devices provide.[2]

Thousands of apps are now available both in Android Google store and Apple app store, and the number of apps being added to the stores is still increasing at an alarming rate. Most of these apps are based on web where they get the contents from web servers. For doing so, they make use of the standard HTTP protocol in order to display the contents from web and to enable users to easily access the web services and interact with the web servers. Most of these web-based apps are designed primarily to support a particular web application. [5]For example, the facebook mobile app is designed particularly for facebook in order to display the facebook content on mobile and also provide easier and better integration with facebook. Therefore, users often prefer to use these web-based apps on their Smartphone.

The technology that enables these web apps to perform the specific web application task such as page rendering, JavaScript execution, etc. is called WebView. WebView is an essential component that is used to display the contents of web services in Smartphone and provides easy and better interaction to the users.[4][5]

1.1 About WebView

WebView is basically a class which is an extension of the Android's View class, which enables to display the web pages. By using WebView, the Android applications can easily embed a browser inside them which not only allows to display the web contents but also to interact with the web servers. There are two types of APIs (Application Programming Interface) in WebView, Web-based APIs and the UI(User Interface) based API. Web-based APIs are used to interact with the web-contents and to access the web services. UI based APIs are the interactive components such as buttons, text fields, etc.[4]

To add WebView in our application, we can make use of the following example:

```
WebView WV = new (WebView) findViewById (R.id.webview);
WV.loadUrl("http://www.example.com");
```

When the WebView is created, the application can load the web page by using loadUrl() method when the Url String is provided. JavaScript is by default disabled in WebView. We can enable it by setting the setJavaScriptEnabled() to true. [4][5]

```
WebView WV = new (WebView) findViewById (R.id.webview);
WebSettings wset = WV.getSettings();
wset.setJavaScriptEnabled(true);
```

2. ATTACK IMPLEMENTATION

In order to launch an attack, following are the three main assumptions:[2][4]

- Before we install any app on our mobile devices, we need to grant the permission for it to be installed completely. In most cases, the application needs to be granted with the android.permission.INTERNET permission which makes an attacker to launch an attack very easily.
- In order to install Android applications in the mobile devices, the user is required to remain logged in to his Google account which is paired with the user's phone. So, the user has to store the login cookies in the browser which also leads to cross-site attacks.

- There are also many free applications which are designed to request for the android. Permission.READ_CONTACT permission and as the applications are granted the permission, they might extract our contact information and other sensitive data. Since the owner of the web content in WebView and the application developer are not the same people, so there could be a potential threat from malicious applications. Examples of such apps are weChat, Line, whatsApp, etc.[6]

3. CROSS-SITE SCRIPTING ATTACK

Cross site scripting attack is also a type of vulnerability which is commonly found in the web applications. It is an injection problem in which an attacker injects malicious code in trusted web sites using web applications. Although, in android, the browsers uses sandboxing mechanism which protects the user from malicious code and limits the scripts to access only the resources available in the origin web site. But this mechanism fails if a user downloads and executes a malicious code from a trusted site unknowingly. So, in such case the malicious script is granted full access to all the resources that are associated with the trusted site[4].

Thereby executing the malicious scripts through malicious applications, an attacker gains full access to the sensitive data and information such as cookies, contacts, location, etc. So, in order to reduce the scope of potential threats, Android must limit the functionality required by the applications from WebView. It has been modeled that JavaScript injection is used for attacking the WebView component using WebView's loadUrl() method.[5] This method receives the argument of type string and if the string starts with JavaScript then WebView treats the entire string as JavaScript and executes it on behalf of the web page that is currently opened in the WebView component. The JavaScript code has the same authorization and privilege as that of the web page scripts and therefore can manipulate the cookies and information on the web page. The attacks can be implemented by executing the code that resides at the server and sending malicious scripts to the server through HttpGet and HttpPost methods which results in stealing cookies and using the stolen cookie to impersonate user.

3.1 Stealing Cookies

A cookie known as a web cookie or http cookie is a small piece of text stored by the user's browser. Cookie is used to store and maintain users authentication and to implement navigation, possibly across multiple visits. Cookie stealing is the most common task in cross-site scripting (XSS) attacks in which the session ID's, Login details of the user is gathered without user's knowledge. These stolen cookies and Url's can be sent to any attacker's server. After the attacker gathers all the cookies, he can easily launch any attack on the user. Such situations can also give rise to blackmailing and threatening the user with his information. Hence, this type of attack is very dangerous because the user is only able to see the trusted site and unknowingly his cookies are stolen. This attack of stealing cookies is shown in Figure 1.

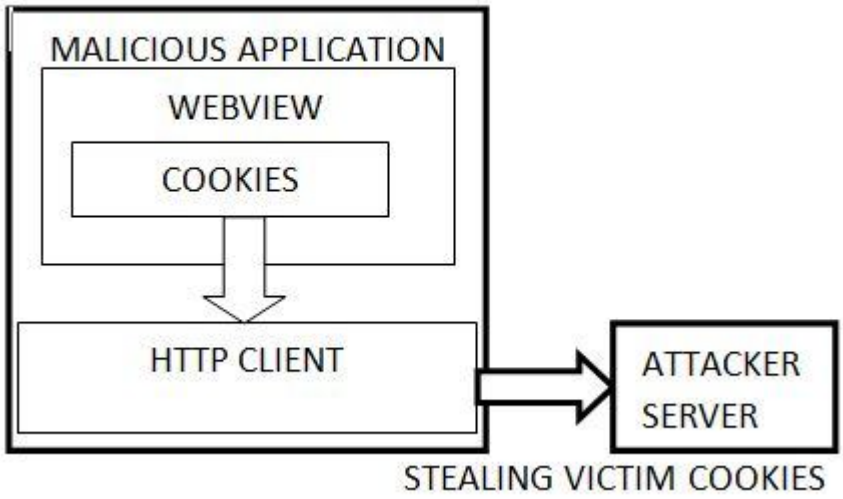

Figure 1. The Attacker stealing cookies from the victim's device.[4]

3.2 Gathering sensitive information

XSS attack is also implemented to extract the sensitive data and information such as email ids, contacts, account numbers, contacts, etc. from user's mobile device. Unfortunately, the user is unaware of these attacks as the user is able to see only the trusted content on the web page. For Example, when a user installs an app, some permissions need to be granted for the app to be installed. Then the malicious app can access and extract all the sensitive information from the victim's device and send them to the attacker's server. A very common and widely used third party app is the facebook. These attacks are therefore very easy to launch and equally difficult to detect. Figure 2. Demonstrates this type of attack.[4]

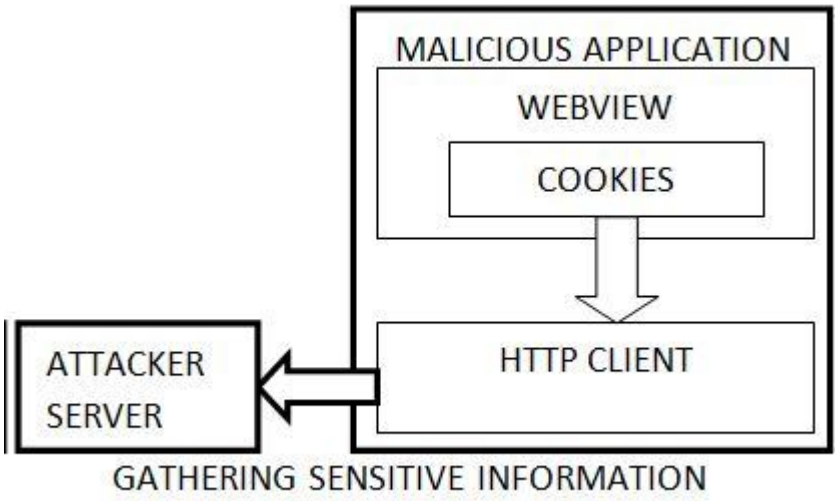

Figure 2. Gathering sensitive data from victim's device.

4. CONCLUSION

The WebView component has enabled the Android apps to add appealing and rich experience to the Smartphone users but at the cost of their security which is very critical. In this paper, we have focused on the implementation of XSS on WebView by injecting JavaScript code. The main cause of attack is unawareness of the user while accessing a web page and installing an app, since the user only sees the legitimate web page and unknowingly runs the malicious code on his mobile device. So such attacks results in stealing cookies and gathering sensitive information. XSS is therefore very easy to launch but very difficult to prevent. Our future work will focus on developing solutions and to defend against these attacks on WebView.

5. REFERENCES

[1] Simon Roses Femerling (2012), Smartphone Apps Are Not That Smart: Insecure Development of Apps in Android Mobiles, Vulnex Research Paper.

[2] Michael Backes, Sebastian Gerling and Philip von Styp-Rekowsky (2011), A Local Cross-Site Scripting Attack against Android Phones, Saarland University, Germany.

[3] Pankaj Sharma, Rahul Johari and S.S Sarma (2013), Combined Approach to Prevent XSS Attacks and SQL injection, SPsymposium-paper.

[4] A B Bhavani (2013), Cross-site scripting Attacks on Android WebView, International Journal of Computer Science and Network.

[5] Tongbo Luo, Hao Hao, Wenliang Du, Yifei Wang and Heng Yin (2011), Attacks on WebView in the Android System, Syracuse University, USA.

[6] Burns (2008), J. Developing Secure Mobile Applications for Android. iSEC Partners, http://www.isecpartners. com/files/iSEC_Securing_Android_Apps.pdf.