Research on android infiltration technology based on the silent installation of an accessibility service

Haitao Sun¹, Chengjie Jin², Xiaohan Helü², Hui Lu², Man Zhang³ and Zhihong Tian²✉

Abstract
This work focuses mainly on the penetration test function and stability of Android smartphones under Kali 2019 and creates a stable penetration function of Android smartphones to form a strong reverse transmission control protocol connection. Using the silent installation technology supported by the Accessibility Service (integrated on a nontoxic functional platform), the Android Package that injects the penetration function successfully bypasses the common mobile phone antivirus software on the market to improve the penetration efficiency.

Keywords
Accessibility service, Android, penetration testing, penetration efficiency stability, reverse transmission control protocol, silent installation, trojan

Date received: 13 October 2019; accepted: 11 December 2019
Handling Editor: Xiaojiang Du

Introduction
Things Net is known as the third wave of the information revolution, and its development will have a huge social and economic benefits. With the successful application of the Internet of Things in all areas, security issues are becoming increasingly apparent. Wireless sensor networks due to play in the task of linking the traditional network of things, the security problem is particularly prominent. Software must be able to define the bridging device in a wireless sensor network, the sensor data collection and display in an intuitive manner information about the end-user, or delivered to the user’s computer, tablet, or smartphone via the Internet is through a display, so our work mainly around Android smartphones launched to analyze the security challenges which it faces with, and the key penetration of these technologies.

From the official Google document, Accessibility Service¹ is a helper class that developers can use to build an accessibility service that meets their needs. The current mainstream technology for silent installation is the pm install technology and source code modification technology for install-package. The former method requires the root permission to be opened, while the latter requires a relatively complicated modification process, and the universality is very low. Therefore, this article discusses how to use this service as much as possible to pop up the system installation interface and

¹Institute for Public Safety Research, Tsinghua University, Beijing, China
²Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China
³Peng Cheng Laboratory, Shenzhen, China

Corresponding authors:
Hui Lu, Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China.
Email: luhui@gzhu.edu.cn
Man Zhang, Peng Cheng Laboratory, Shenzhen 518000, China.
Email: zhangm@pcl.ac.cn
install the installation package with the infiltration function program to improve the penetration efficiency.

Penetration testing is a technical means of imitating hacker attacks and exploiting vulnerabilities. Through a series of security tests on penetration, the security performance of the target system is detected, and the vulnerable nodes in the analysis system that are unresolved or have not been discovered before are identified. At present, the development of penetration technology is becoming increasingly mature, and the rapid development of mobile Internet technology has increased security demands. Among this technology, mobile smart terminals, especially open source Android smartphones, can easily become targets of hackers. Through this research, Metasploit has a good integration degree for Android penetration technology in the Kali environment, but its stability has certain defects. Therefore, it is necessary to carry out stability improvement research.

Our evaluation shows that our method could achieve a good stability effect. Specifically, our article makes the following contributions.

We propose a source-level optimization method to modify the Metasploit reverse transmission control protocol (TCP) connection with the application startup service script together to increase the stability of penetration testing.

Our silent installation technology does not require the device to be rooted. Unlike other methods, it is a lightweight operation that is easy to implement.

Because our application inherits the Accessibility Service, it does not require sensitive mobile phone permissions, and it will not be detected by anti-virus software in the Appstore.

**Overview**

**Accessibility service**

Accessibility Service is designed by Google to help disabled people who cannot use Android devices and apps. This service runs in the background. When the mobile smart terminal system changes, represented by AccessibilityEvent, the monitor monitors the interface closely. Once the interface changes, the code can be accessed by callback. For example, when the system installation interface appears on the screen through a pop-up window, we automatically click the “Install,” “OK,” or “Next” button by simulating the user’s finger through Accessibility Service.5

In particular, the implementation of this method needs to inherit the AccessibilityService class, which needs to implement the two core methods onAccessibilityEvent and onInterrupt(). There are many ways to use this class, such as finding a control of a particular focus type, returning the root node of the currently active window, obtaining system services, and allowing the service to wait for keystrokes.

**Android system**

Since 2018, domestic mobile phones such as Oneplus, Xiaomi, and Huwei have existed on the world stage; their common features are based on the Android system.4 Android is a free operating system with open-source compatibility, strong plasticity, high researchability, and a low research threshold. In recent years, Android smartphones have maintained the top position in the market share, which has made Android the most popular open-source project. As of the second quarter of 2019, the Android system has a market share of more than 87.4% for global mobile intelligent terminal systems, ranking first in the world. Studies on this topic have also proliferated dramatically. At the same time, the attacks on these devices are not to be underestimated. The scale of attacks by hackers is still very large and has never stopped, and the situation is in a state of dynamic balance.

**Penetration testing**

Penetration testing is a test method that can be divided into black-and-white box testing, gray box testing, and implicit testing. The main methods are infiltrated into the target system. There are three main types: port, operating system detection, and firewall intrusion detection. First, external behavior detection consists of access control and so on. Second, we have vulnerability detection by software and server vulnerability testing, common vulnerabilities and exposures (CVE) level binary vulnerability testing, Web scripting, and application testing. Third, distributed denial-of-service (DDoS) attacks, VPN attacks, address resolution protocol (ARP) spoofing tests, internal behavior attack node detection consisting of cache poisoning, session eavesdropping and hijacking tests, password and password decryption, social worker tests, and backdoor tests, and so on fully reveal the security problems and vulnerabilities of the system itself, combined with the test content of extensive penetration testing on mobile intelligent terminals. The operating system conducts a comprehensive and detailed study to systematically discover vulnerabilities in the mobile operating system. This work is of great significance for preventing malicious attacks by hackers and maintaining the security of a mobile Internet of Things operating system.

**Improving penetration stability**

With the rapid development of infiltration technology, the penetration test content and its functions integrated on Kali are constantly enriched and improved, but such
a series of attacks must first be established in a stable session, that is, the reverse TCP connection session is stable on the premise of continuous operation and a guarantee to not drop the line. At the same time, to meet the security requirements of mobile intelligent terminals, the bandwidth of this connection should not be set too large, and the memory occupancy rate also merits attention. The hidden problem is the most typical problem to be solved. The stability of the network condition of the mobile terminal also affects the quality of the reverse TCP session to some extent. Therefore, increasing the stability of the session in various aspects becomes an important criterion and standard for evaluating the stability of the penetration attack.

**Silent installation design with accessibility service**

**The design architecture of silent installation**

Accessibility is also called “Hook technology” in Android. In Windows, the role of Hook is to be able to listen to any content of interest, and AccessibilityService in Android can also listen to some of the features we need. The accessibility-based silent installation implementation architecture is designed for four phases: AccessibilityManagerService configuration, AccessibilityService Info configuration, user interface (UI) exploration, and AccessibilityManager. The architecture diagram is shown in Figure 1.

**AccessibilityManagerService configuration.** Due to the permissions required for AccessibilityService, we use the BIND method with configuring filtering directly in meta-data so that the system can bind this service (Figures 2 and 3). This is an effective way to minimize the amount of redundant information.

**AccessibilityService Info configuration.** PackageNames is the package name of the corresponding application monitored by Accessibility Service. If there is a requirement, it is possible to listen to multiple applications’ communications packages. In the source code analysis of Android, the package names of most installers appear, and the appropriate package name needs to be found. The core content of the silent installation is automatic installation. Its principle is as follows: first, the node object instance of AccessibilityNodeInfo can be obtained using the getRootInActiveWindow() and getSource() methods, and this instance can be used as the UI node of this event. Then, we rewrite the AccessibilityService class to construct an onAccessibilityEvent method that listens to changes in the mobile smart terminal interface and triggers the callback function. Therefore, the following points should be addressed:

- Set the event that responds to an application. The default setting is to respond to all application change events; to listen to multiple packages, set the string[] parameter;
- Analyze the type of response event, such as a click, long press, or slide. If it is necessary to specify a response for all events, the type is written as TYPES_ALL_MASK.
- If the value of canRetrieveWindowContent is set to false, the object obtained by getSource() is null, which is important because the content of the window cannot be obtained.

**Exploration of interface UI elements.** The findAccessibilityNodeInfosByText method can obtain the control node based on the text content displayed by the control. Note that the logic of the method is “contains” rather than “equal to”; in other words, the...
method supports fuzzy lookups and not necessarily exact lookups. The method looks for "install" text, and then it looks for “Is it installed?.” The system title “install” for these methods will be found, so we need to determine whether it is the text content on the control we need. Finally, we use the simulated user to click the “OK,” “Install,” “Next,” and “Finish” buttons (List 1) to achieve the effect of automatic clicks.

Different mobile phones will have a variety of different content at the installation interface, and there are many vocabulary controls that are not “installed” but synonymous. Therefore, the content definition can be increased or decreased according to the needs. To obtain better Android mobile phone compatibility and universality, it is necessary to fill in more and better text content for a wide range of coverage.

Delivery in AccessibilityManager. Before the triggering event is executed, it is determined whether the UI element still exists on the interface, and the accessibility event listener immediately starts receiving the auxiliary event after the detecting step. After obtaining the UI element on the current operation interface of the mobile intelligent terminal, the action execution method is used to send an operation effect request to the smartphone. At this point, the accessibility service system began to enter the cycle phase and achieved the effect of silent installation. In the AccessibilityManager’s sendAccessibilityEvent() function, the corresponding IBinder object is obtained through the ServiceManager, and then a service instance is converted from the IAccessibilityManager. In this process, the sendAccessibilityEvent() function forwards the listened event, and IAccessibilityManager is retrieved from the forwarding process by the getServiceLocked() function.

In the process of information transfer, the task of detecting the corresponding service object is completed by the getServiceLocked() function, and the tryConnectToServiceLocked() function is called if the detected value is empty. The function of ServiceManager is the core of the entire Accessibility system. This component also plays a pivotal role in the Android system. ServiceManager is run immediately after the initialization process starts and is used to manage various systems that are more common in Android systems. ServiceManager levels services...
(InputMethodManager and ActivityManagerService are also subject to it) and provides the client with the ability to query related services. After obtaining the service object, AccessibilityManager lets the system receive the auxiliary event message and then uses the message package name to detect the related AccessibilityService. After the detection step, the onAccessibilityEvent() function immediately begins receiving the auxiliary event. At this point, the accessibility system begins to enter the cycle phase and achieves the effect of silent installation.

Turning an ordinary Android Package into an effective penetration tool

This step is to use the Android de-compilation technology to modify the normal application installation packages on the market. For example, a “flashlight” installation package decompiles the main function and startup entry keywords in the Manifest.xml file and precisely locates the statement that starts the application package (Figure 4).

After the specific modification process to find the onCreat method in the Smali file, the bundle instance is the point of focus, and we must add the load code in charge of the startup. Once the file is confirmed to be saved, we move the Metasploit folder of the payload to the com directory under the Smali of the flashlight, and we then return to the xml file to add some permissions for the infiltration and compile it back. The last process is signature authentication and the wait for the test machine to download and perform a silent installation to improve the efficiency of the penetration work.

The improvement of permeability stability under Kali

Metasploit is one of the most popular penetration tools for open-source tools, helping security experts identify security risks and validate vulnerabilities. This infiltration tool has a wealth of features, including exploits, web script scanning, and social engineering. Metasploit is now integrated into the Kali operating environment. There are as many as 34 attacks and exploit modules for Android. Currently, there are many scenarios of penetration attacks without roots, such as hidden photos, SMS content acquisition, forwarding, SMS, map location, and hidden icons. However, through many tests, it has been found that the reverse TCP connection session is not very stable and that the switching and background placement times are too long to affect the session robustness.

Stability improvement based on the source code

After many tests, the connection established by the Metasploit Android payload on Kali has been found to be erratic. Many researchers worldwide have provided a variety of solutions. The most common method is to execute a shell script after the connection is established. However, this solution requires that the connection is established successfully and remains connected. In recent years, we have deepened the research on Metasploit’s Android payload and found that the payload source code is modifiable, which offers a more targeted and more effective way (List 2) to solve the root cause than other approaches.

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Context start Accessibility service</td>
</tr>
<tr>
<td>2.</td>
<td>Rewrite the onBind method of the IBinder class</td>
</tr>
<tr>
<td>3.</td>
<td>Trigger with intent</td>
</tr>
<tr>
<td>4.</td>
<td>return NULL</td>
</tr>
<tr>
<td>5.</td>
<td>Rewrite the onStartCommand method</td>
</tr>
<tr>
<td>6.</td>
<td>Trigger with intent</td>
</tr>
<tr>
<td>7.</td>
<td>If Payload</td>
</tr>
<tr>
<td>8.</td>
<td>return the value of start stick</td>
</tr>
</tbody>
</table>

List 2. Startup service section of the Accessibility Service source code improvement.

The code shows that there is a start method in the service startup, which is the core code of the execution payload. Then, if one wishes to achieve a sustainable connection to the load, one needs to know how to obtain the start method of the service and load the running process continuously. After research and investigation, we ultimately chose to add the following features:
Automatic restart function when the service is destroyed; Service timing execution function based on the alarm management method.

The onDestroy method (List 3) in the rewrite service class can be used to implement the automatic restart function when the service is destroyed. At the same time, locate the Metasploit source code that contains the Android attack and the module that uses the module, namely, the MainService.java package, and add the following code to the MainService class:

```
1. Rewrite the onDestroy method:
2. New Local Intent
3. Set Local Intent(MainService)
4. Startup Service(Local Intent)
```

List 3. The opening part of the source code that results in automatic destruction.

After the actual measurement, after this code is added, when the app is closed on the Android phone, the reverse TCP connection is re-established when disconnected. However, this method is still not perfect. Payload disconnects the service and does not end. It does not call the onDestroy method, and the service is not re-executed. Therefore, we choose to make the service execute regularly by adding AlarmManager. In addition, in the MainService class, we modify the onStartCommand method (List 4):

```
1. Rewrite the onStartCommand method
2. Startup Payload
3. Create a manager object of the AlarmManager class
4. Set restart time
5. Set service time = system real time + restart time
6. New Metasploit event
7. Set the object of getBroadcast pi
8. AlarmManager.ELAPSED_REALTIME_WAKEUP method to set Manager
9. return START_STICKY value
```

List 4. Startup command section of the improved source code.

In the above code block, we can modify the restart time as needed. Greatly Integrated Metasploit provides a MainBroadcastReceiver class as a dedicated receiver for AlarmManager; However, we found that this receiver only receive ACTION_BOOT_COMPLETED in many cases. Thus, our solution (List 5) is to modify the broadcast receiver file (which contain stage/MainBroadcastReceiver.java) to remove the evaluation code directly.

```
1. Rewrite the onReceive method
2. MainService startup context service
3. Delete evaluation
```

List 5. Receiving parts of the Accessibility Service source code improvement.

In addition, modify the corresponding configuration of the receiver in AndroidManifest.xml. And in the process of compiling, one needs to pay attention to the following four points:

1. Use Android SDK, NDK, and Maven.
2. Download the Android SDK with version numbers 10 and 19.
3. Look for the Java directory variable in the environment variable, and compile it on its path. When the Android payload for Metasploit displays “compiled successfully,” the modification is completed.
4. Finally, find the file with the Android type in the target file path, and copy it to the data directory specified by the file name Metasploit-framework.

Using a shell script to improve stability

It should be noted that when writing the shell script file to the Linux environment, our solution is to retype the line break in the Linux environment because where Windows has a carriage return + line feed (\r\n), Linux has a newline command (\n), and the following scripts are used (List 6):

```
1. While true
2. do startup --user 0 -a Corresponding application action.Main -a
3. Corresponding application ‘s the MainActivity of com package
4. sleep 20
5. Done
```

List 6. Persistent shell scripts.

The corresponding implementation method\textsuperscript{19} is to restart the android.intent.action.MAIN every 20 s, and the MainActivity method in the package file of the corresponding app is also followed by the emphasis on loading once. The script simulates the process of user action as restarting every 20 s and loops with “Starting:Intent.”

Table 1 shows the number of consecutive TCP connections before and after the improvement in the experiment with a total of 500 tests (Figure 5).
Through the analysis of the results of the above experiments, it is found that the quality of source code modification is generally better than that without modification, and the script modification can also play a role in improving stability. The two modification methods can be used together, which increases the stability to close to 92% because the source code remains stable and the script ensures subsequent operations. Therefore, the conclusion is that the two methods work best together.

Related work

The first malware to exploit ADB resource exhaustion was DroidDream, which sends malware to Android devices by installing a rootkit on the device. By contrast, our installing does not rely on ADB. TouchLogger exploits the side channel to infer keystrokes; SimBad is hidden in the Google Play Store for users to download and install.

However, many attacks have certain aggression. What we are different is that we inherit the Accessibility Service class, which is used by Google to expand the market. Therefore, it has a broad application prospect and is highly credible. We have developed a kind of method within this trend and improve its stability instead of other constraints.

Conclusion

With the rapid development of information domain expansion, new types of attacks against the mobile information Internet of Things have begun to proliferate, threatening network security. The increasingly sophisticated and versatile hacker attack is an undeniable reality. Only by “knowing ourselves and knowing each other” can we first understand the attacker’s ability, intention, and means to take the lead in action; build targeted protection in advance; and take measures to avoid causing disasters. Therefore, fixing security vulnerabilities to defend against cyber attacks is imminent.

With the rapid rise of the mobile Internet of Things and the Internet, both manual auditing and automated infiltration techniques need to be built on technologies that remain stable in penetration testing sessions. This article implements the silent installation of Android Package (APK) through Accessibility Service. This installation can make the APK with its permeation function easy to install on the mobile phone and realize the automation effect. Then, we let the infiltration program obtain the control right of the smartphone, and the penetration efficiency is greatly improved. In addition, this article studies a variety of stability enhancement methods for reverse TCP connection sessions. This study is valid for most Android systems and contributes greatly to improving the quality of penetration testing. In the future, we will continue to follow up on the security improvements of the Android operating system, propose more effective analysis methods, and find and propose feasible solutions to protect the security of mobile intelligent terminals.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This paper was supported by the Guangdong Province Key Area R&D Program of China under Grant No. 2019B010137004; the National Natural Science Foundation of China under Grant Nos 61972108, U1636215, 61572153, and 61972108; and the National Key Research and Development Plan under Grant No. 2018YFB0803504 and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019).

ORCID iD

Zhihong Tian https://orcid.org/0000-0002-9409-5359
References


