DOPA: Detecting Open Ports in Android OS

Ali Alshehri*, Hani Alshahrani*, Abdulrahman Alzahrani*, Raed Alharthi*, Huirong Fu*, Anyi Liu* and Ye Zhu†
*Oakland University, Rochester, Michigan USA
†Cleveland State University, Cleveland, Ohio USA
Email: {aaalshehri, hmalshahrani, aalzahrani, rsalharthi, fu, anyiliu}@oakland.edu

Abstract—In smartphones, open ports can be exploited by malicious applications to allow attackers to remotely control user’s device. In this work, we perform an attack that opens a port on an Android device to remotely connect to this port and access various phone features without requesting any permission from the user. Additionally, we develop and implement a tool called, Detecting Open Ports in Android operating system (DOPA), that detects open ports and reports them to the user.

I. INTRODUCTION

Among different smartphone platforms, Android operating system (OS) has become the most prevalent and fastest growing platform. The popularity of Android has encouraged applications’ developers to create more applications to serve Android users. Google Play is the official Android application store for the Android OS, where users can explore and install applications from different categories. Moreover, users can download applications from third-party stores. These stores including the official market are targeted by adversaries to inject their malicious applications. According to Andrew [1], Google removed more than 700,000 applications from the official market only during 2017.

The security of Android system was designed with multiple layers to deliver flexibility, especially for the developers to build secure applications, and sufficient protection for all users. Android OS provides different security mechanisms to protect data, applications and users. One of these mechanisms is permission model which is used to protect the communication between applications and the Android system. Any application that wishes to communicate with another application, accesses public resources (e.g. camera, Bluetooth, etc.) or highly privileged APIs (e.g. read or send SMS) must have the appropriate permission. However, permissions are not fine-grained enough, which enables an unscrupulous developer to get unauthorized access to device information.

One of the vulnerabilities in Android OS is using open ports, which does not require any permission and can be misused by some unscrupulous developers. Open ports can be exploited by malicious applications as a backdoor attack to allow attackers to remotely connect to these ports through an application or web page script and control over a user’s device due to inadequate protection [2]. With this poster, we perform an attack that opens a port on an Android device to remotely connect to this port and access various phone features without requesting any permission from the user. To mitigate this problem, we develop and implement DOPA, a new approach to detect open ports used by non-system applications on Android devices.

II. ATTACK MODEL

As shown in Figure 1, an unscrupulous developer could develop an application that has the ability to access user’s device via Secure Shell (SSH) commands. This application might include a hidden malicious code and can be distributed through the official applications market or any other third-party applications markets. Once this application is downloaded by end users on their devices, it will open one of the device’s ports and send the device IP address to a database managed by the attacker. The device IP address along with the predefined port number can be used by the attacker to remotely access the user’s device. As a consequence, a hidden channel connection with the victim device can be established without the user consent, which allows the attacker to send or read the user’s sensitive information.

III. RELATED WORK

The aforementioned attack is a new threat in mobile platforms that is not explored by many researchers. At the time of writing this poster, only two other research discussed this threat in Android OS. First, Qusay et al. [3] demonstrated this type of attack by developing a wallpaper application that gives an attacker a shell access to a victim’s device using netcat Linux utility tool. This work does not implement any solution, but suggests to re-evaluate the permission model of the Android OS by Google. Second, Yunhan et al. [2] investigated the usage of open ports by designing a static analysis tool called, OPAnalyzer. They proposed an API called SecureServerSocket that authenticates the open port usage through a 2-Step Verification technique. However, this solution is not comprehensive since the proposed API should be considered by applications’ developers when there is a need to communicate through an open port.
A major drawback of these works is that the usage of open ports remains vulnerable and users are not aware of any application using them. Therefore, our contribution of this research is to monitor all requested ports and keep users aware of all detected ports to take the proper action.

IV. PROPOSED METHOD

To detect open ports on Android OS, a tool called DOPA has been designed and implemented, which monitors all ports used by all installed applications and report them to the user. All applications running on the background are categorized into system applications and non-system applications. In our approach, we consider the non-system applications, which are installed by the user either from Play Store or any third-party market. All open ports used by non-system applications are monitored in real time using Network Statistics (netstat) utility tool. This tool provides valuable information about all incoming and outgoing network connections, such as the applications involved in the network communications, the open ports used by applications, protocol types, etc.

As illustrated in Figure 2, DOPA works as follows: First, DOPA will be installed on the system portion of the user’s device. Second, once DOPA is installed, all open ports will be detected by DOPA. These ports will be reported to the user along with application’s name that are trying to use them. Third, if the user decided to keep the port open then the application that opens the port will be allowed to exchange data with a remote server using the IP address of the device along with the port number. However, if the user decided to close the port, DOPA will close the application that opens the port to disconnect all communications with the remote server.

V. IMPLEMENTATION AND EVALUATION

DOPA has been implemented and evaluated on a real Android device to detect all open ports. First, we implement the attack model shown in Figure 1 and we successfully control the Android device through SSH commands. The SSH command was used to browse user’s files and directories. Moreover, we were able to read all text files on the device through the command lines. This is achieved by developing an application that opens a port on Android device. Once this application is installed, the IP address of the device will be sent to our server.

Second, DOPA has been evaluated based on the developed attack model. It was able to detect all open ports on the device. Furthermore, the detection technique provides the user with all detected ports used by other applications and allows the user to stop any suspicious application or keep the application running. Figure 3a shows the main screen of DOPA; while Figure 3b shows the message appears to the user when any open port is detected.

VI. CONCLUSION

Open ports in Android smartphones can be exploited by cybercriminals to remotely control users’ devices and steal sensitive information. They can also allow attackers to inject malicious applications in users’ devices without their consent. In this poster, an attack model has been performed to identify the risks of open ports. Also, DOPA has been introduced, which detects open ports in an Android device and reports those ports to the user.

ACKNOWLEDGMENT

This research work is partially supported by the National Science Foundation under Grants CNS-1460897, DGE-1623713, and DGE-1723707. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES