Bad Battery: Accessing Data and Injecting Malware into Android Devices

Nathalie Domingo
Electrical and Computer Engineering
Carnegie Mellon University
ndomingo@andrew.cmu.edu

Bryan Pearson
Computer Science
Stetson University
peabryan95@gmail.com

Kelvin Ly, Kaveh Shamsi, and Orlando Arias
University of Central Florida
rangertime@knights.ucf.edu, kaveh@knights.ucf.edu, and oarias@knights.ucf.edu

Dr. Yier Jin and Dr. Shaojie Zhang
University of Central Florida
yier.jin@eecs.ucf.edu and shzhang@cs.ucf.edu

Abstract

The main goal of this project was to exploit the vulnerabilities associated with the access given to Android phones through a USB connection. This was to be done without the users knowledge by concealing the malicious hardware inside a portable power bank, and thus to the user it would only appear as if their phone was being charged.

In order to cause both short term and long term damage, two attacks were implemented. The first attack, referred to as Data Access, is aimed at copying information off of the phone and sending it to a designated email address to be viewed by the attacker. While the second attack, referred to as Malware Injection, is aimed at installing a malicious app onto the user's phone that will remain there as long as the user does not remove it.

Background

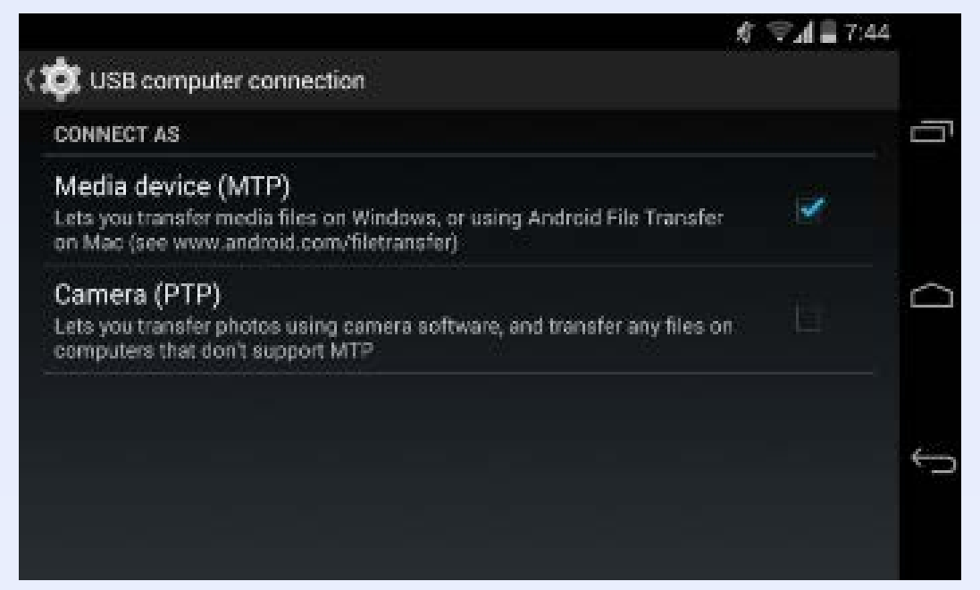

7:44
USB computer connection
CONNECT AS
Media device (MTP) ☑
Lets you transfer media files on Windows, or using Android File Transfer on Mac (see www.android.com/filetransfer)
Camera (PTP) □
Lets you transfer photos using camera software, and transfer any files on computers that don't support MTP

Figure 1: Android USB computer connection settings

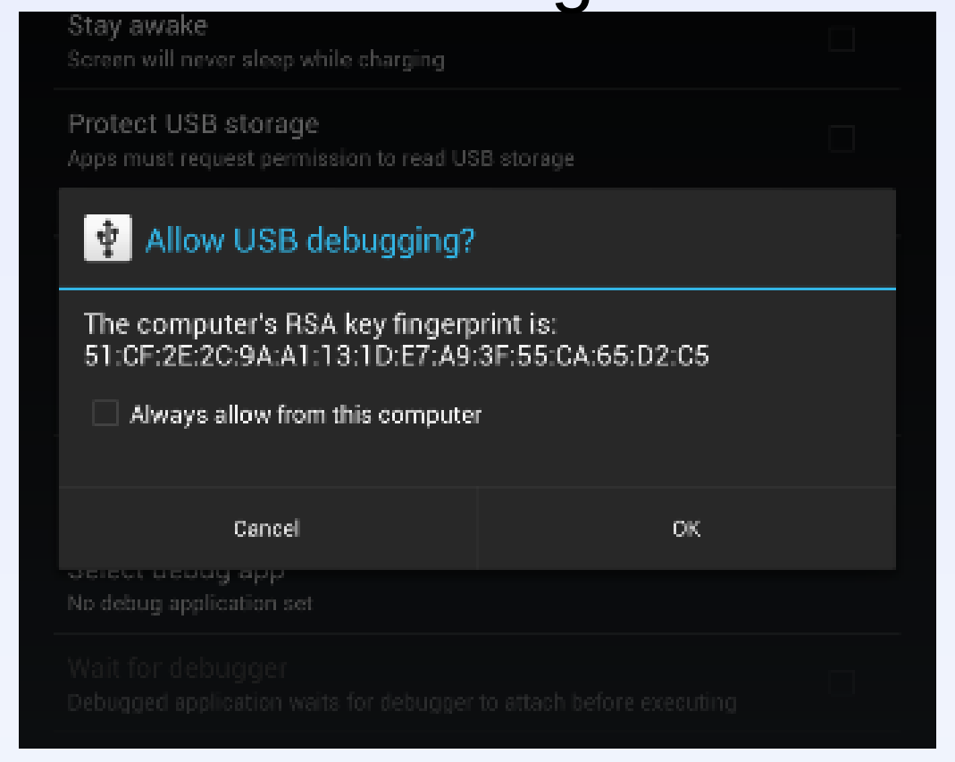

Stay awake
Screen will never sleep while charging
Protect USB storage
Apps must request permission to read USB storage
Allow USB debugging?
The computer's RSA key fingerprint is:
51:CF:2E:2C:9A:A1:13:1D:E7:A9:3F:55:CA:65:D2:C5
□ Always allow from this computer
Cancel OK
Select debug app
No debug application set
Wait for debugger
Debugged application waits for debugger to attach before executing

Figure 2: Computer attempting to connect to an Android device via Android Debug Bridge

Media transfer protocol (MTP): Android setting where the phone is seen as a media device

Photo transfer protocol (PTP): Android setting where the phone is seen as a camera

USB Debugging: Android setting that allows greater access to the phone through a USB connection

Computer's RSA key fingerprint: sequence of bytes to identify the computer

Hardware Components

A Raspberry Pi 3 model B was used as the malicious hardware that runs two scripts to implement the two attacks described above

Figure 3: Raspberry Pi 3 model B

A Random Order power bank was used to power the Raspberry Pi

Figure 4: Random Order power bank

Implementation

The Data Access attack is carried out via a Bash script and is run automatically after the Pi finishes booting up.

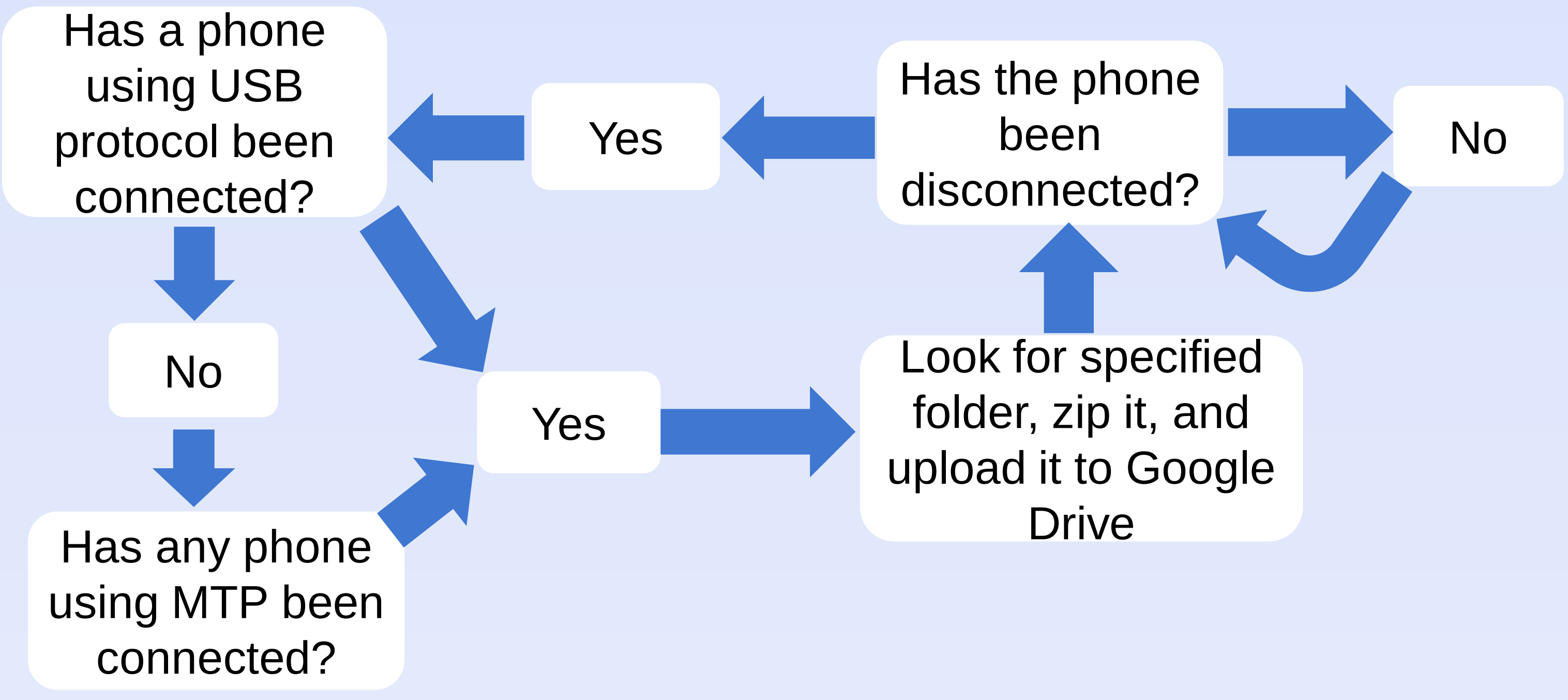

Has a phone using USB protocol been connected?
Yes
Has the phone been disconnected?
No
No
Yes
Look for specified folder, zip it, and upload it to Google Drive
Has any phone using MTP been connected?

Figure 4: Data Access attack process

The Malware Injection attack is carried out via a Python script and is run automatically while the Pi is booting up through Cron, a task scheduler.

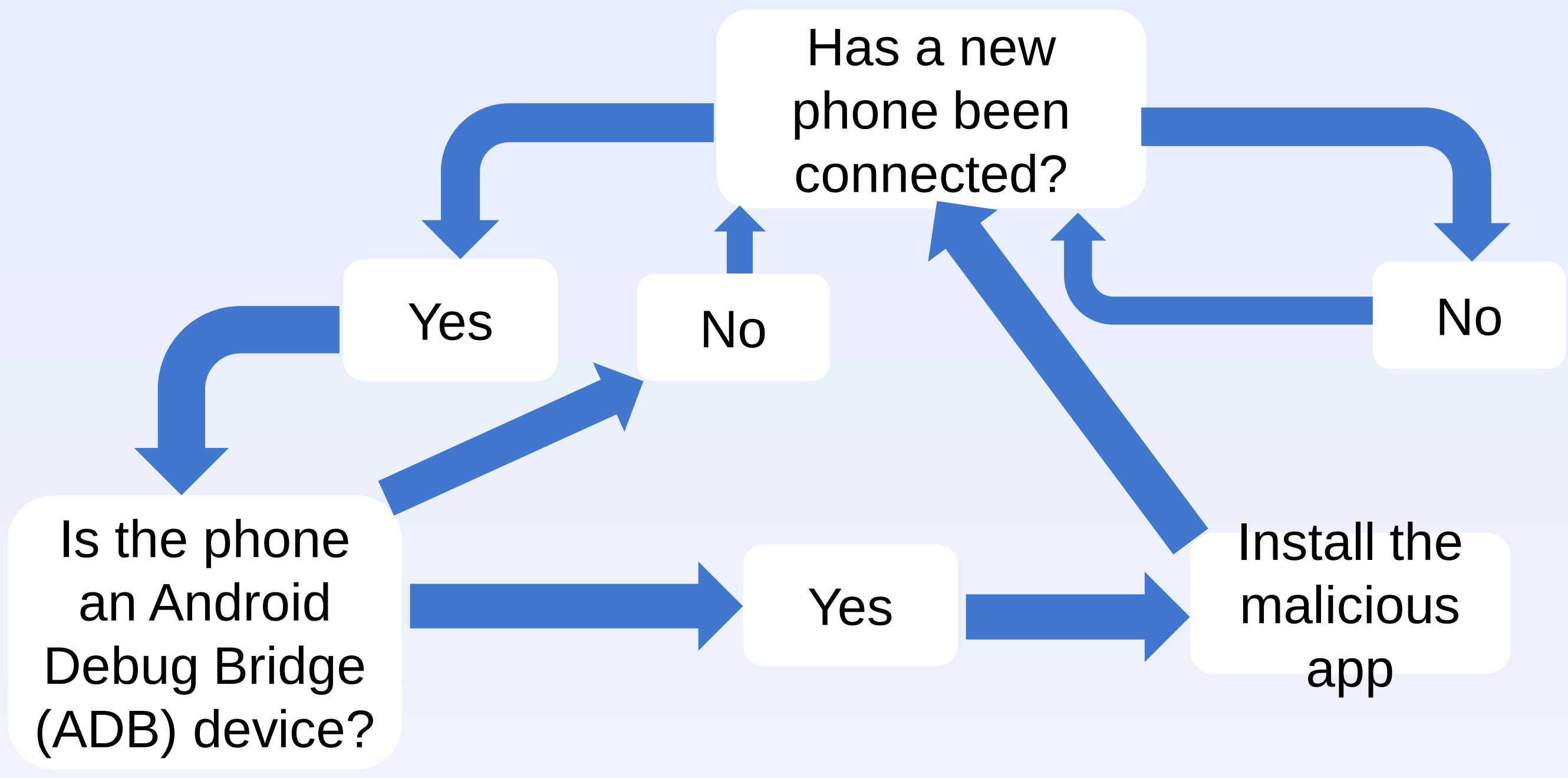
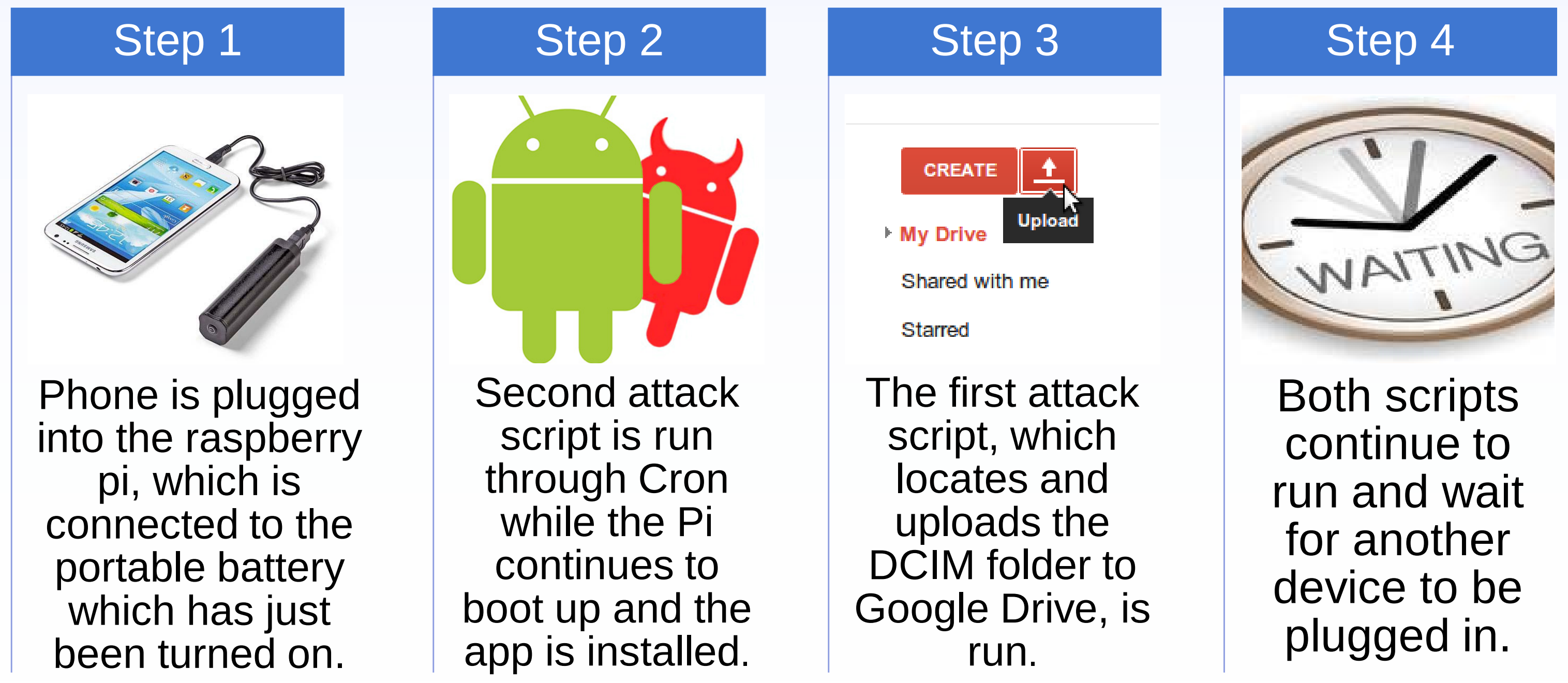

Has a new phone been connected?
Yes
No
No
Is the phone an Android Debug Bridge (ADB) device?
Yes
Install the malicious app

Figure 5: Malware Injection attack process



Step 1: Phone is plugged into the raspberry pi, which is connected to the portable battery which has just been turned on.

Step 2: Second attack script is run through Cron while the Pi continues to boot up and the app is installed.

Step 3: The first attack script, which locates and uploads the DCIM folder to Google Drive, is run.

Step 4: Both scripts continue to run and wait for another device to be plugged in.

Figure 6: Step-by-step explanation of the overall implementation

Results

The Bad Battery worked on all the phones we tested with varying required phone settings and input from the user in order for the attacks to work.

	Data Access Attack		Malware Injection Attack		
	Allow USB Computer Connection From Pop-Up	MTP Enabled	RSA Authentication	USB Debugging	PTP Enabled
Nexus One—Android Version 2.3.6	✓		✓	✓	
Motorola Moto E—Android Version 4.4.4		✓	✓		✓
Samsung Galaxy S5—Android Version 6.0.1		✓	✓		

Figure 7: A chart showing the settings and user input required for the attacks to work

We expected the attacks to require different settings and input depending on the Android version, but we did not expect for the Malware Injection attack to work so well on the newest model. In order for the attack to work, it only needed RSA authentication from the user, but we expected it to also require the phone to have PTP enabled.

Future Work

In order to expand and improve this project, the malicious app injected could be created such that it encompasses the Data Access attack. If the two attacks were combined into one, it would minimize the amount of user input, and would allow for more control over the phone. Additionally, the Data Access Attack could be expanded to also affect iPhone.

Acknowledgements

The support for this work was provided by the National Science Foundation REU program under Award No. 1560302. Any opinions, findings, and conclusions and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation