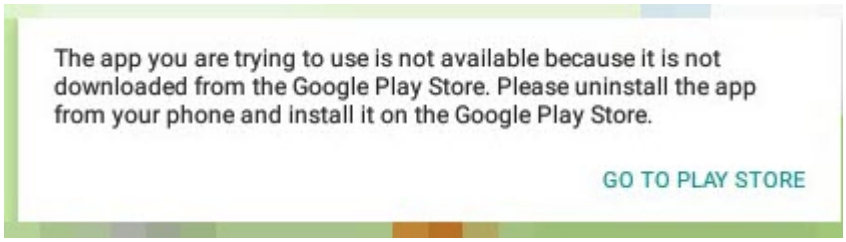
Hooking on Android

written by Mert SARICA | 1 January 2021

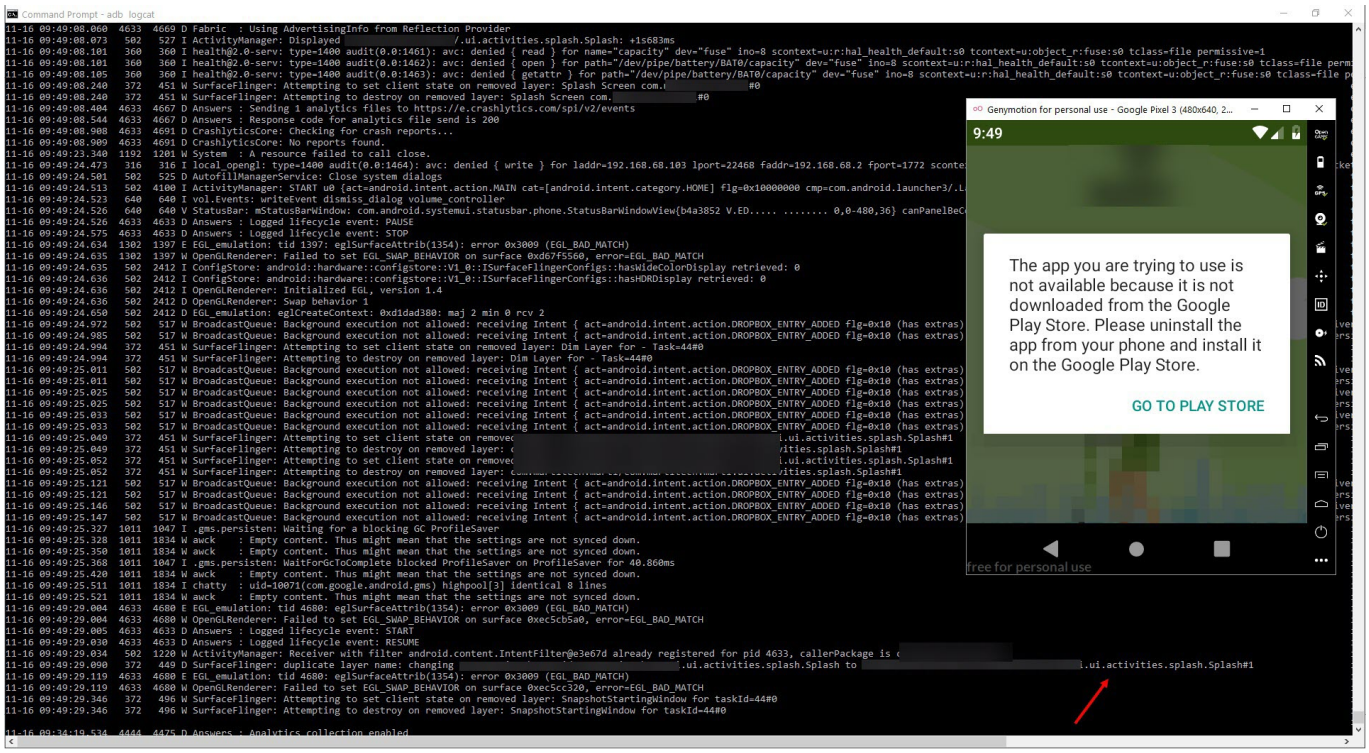
Although our topic is the Android world, when it comes to hooking, I first think of the illegal electricity that is drawn by hooking onto energy transmission lines and into homes. In the Android world, we also use a similar method when we want to dynamically analyze or intervene in applications. So why do we need this? Sometimes, when we want to do a cybersecurity research on Android applications or during a penetration test, we need to analyze the target Android application to find security vulnerabilities. To do this, we usually start by converting the target Android application into source code and doing static code analysis. However, in today's Android applications, the codes are hidden (obfuscation) and made difficult to understand, so we try to analyze the application dynamically using emulators such as Genymotion. The reason for saying "try" is that today, mobile applications like Snapchat apply many methods to prevent dynamic analysis in addition to static analysis.

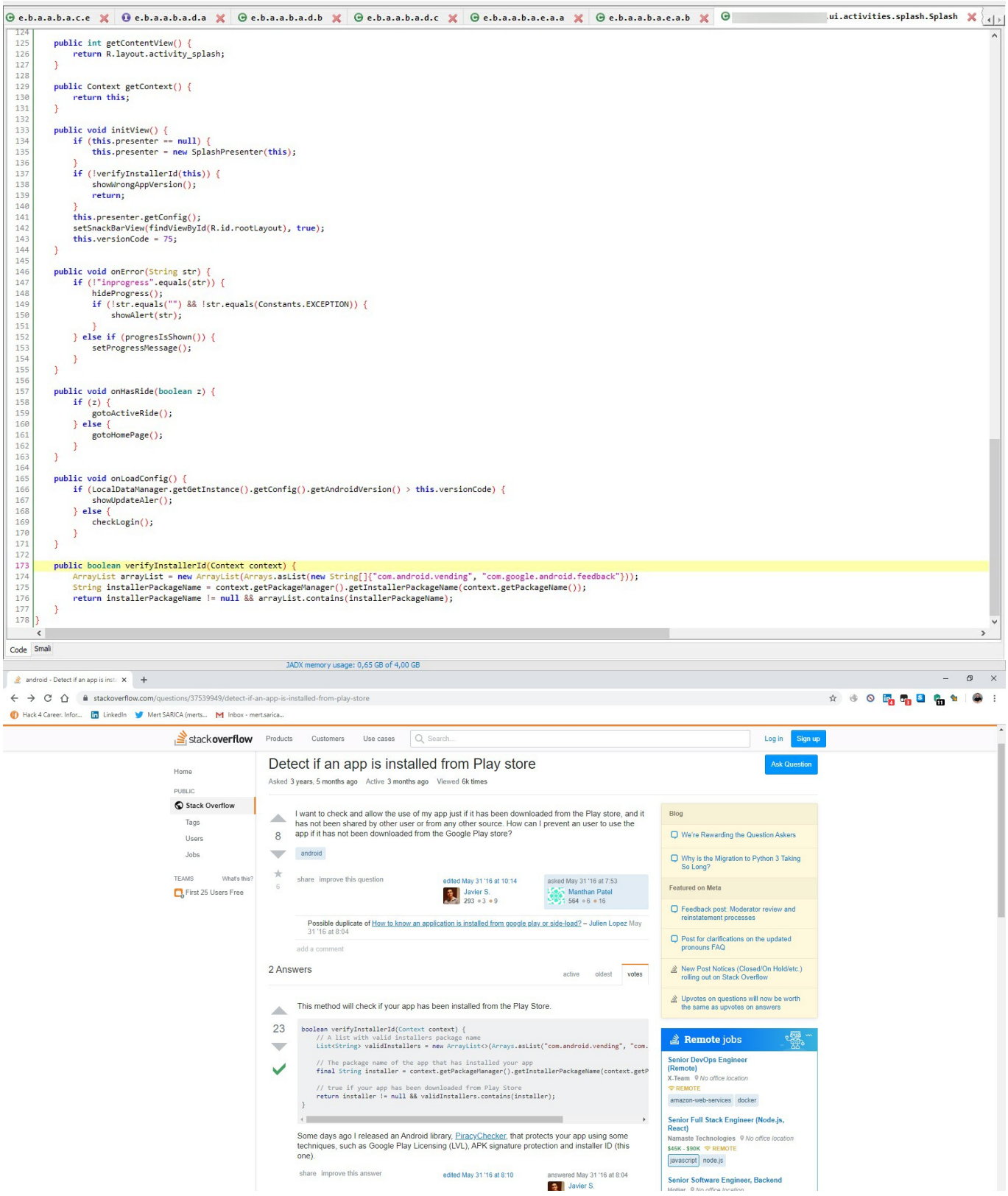
During my trip to Stockholm in December 2019, as I was exploring the city, I noticed that people were passing by me on stylish electric scooters. Since their use is extremely widespread in Europe, I remembered that some security vulnerabilities had been detected in electric scooter applications over time as they began to catch the attention of security researchers. As electric scooters and their applications are also starting to become popular in my country, I decided to take a look at one of the Android applications from a security researcher's perspective, in order to satisfy my curiosity before using them. Of course, as always, reality did not meet my expectations and due to the obstacles I faced, this blog post came about. :)

As a security researcher with limited time, I did not want to waste time with static code analysis, so I downloaded the APK file using the APK Downloader web application and loaded it into the GenyMotion emulator. After running the application, I was surprised to see a warning message that the APK file had not been downloaded from Google Play. :)



So, before diving into blind static source code analysis, I ran the adb logcat command to view system messages on the command line and then ran the application again. I used the jadx tool to convert the APK file to source code and began examining the activities.splash.Splash file to find the function that I thought was related to the warning message that appeared on the screen. At the end of the file, I noticed the verifyInstallerId function, which immediately caught my attention. When I searched for this function on Google, I saw that it was used for this exact purpose. This function used the Android's getInstallerPackageName function to check if the application was installed by Google Play. If the application was installed by Google Play, the installerPackagename variable would have a non-zero value.

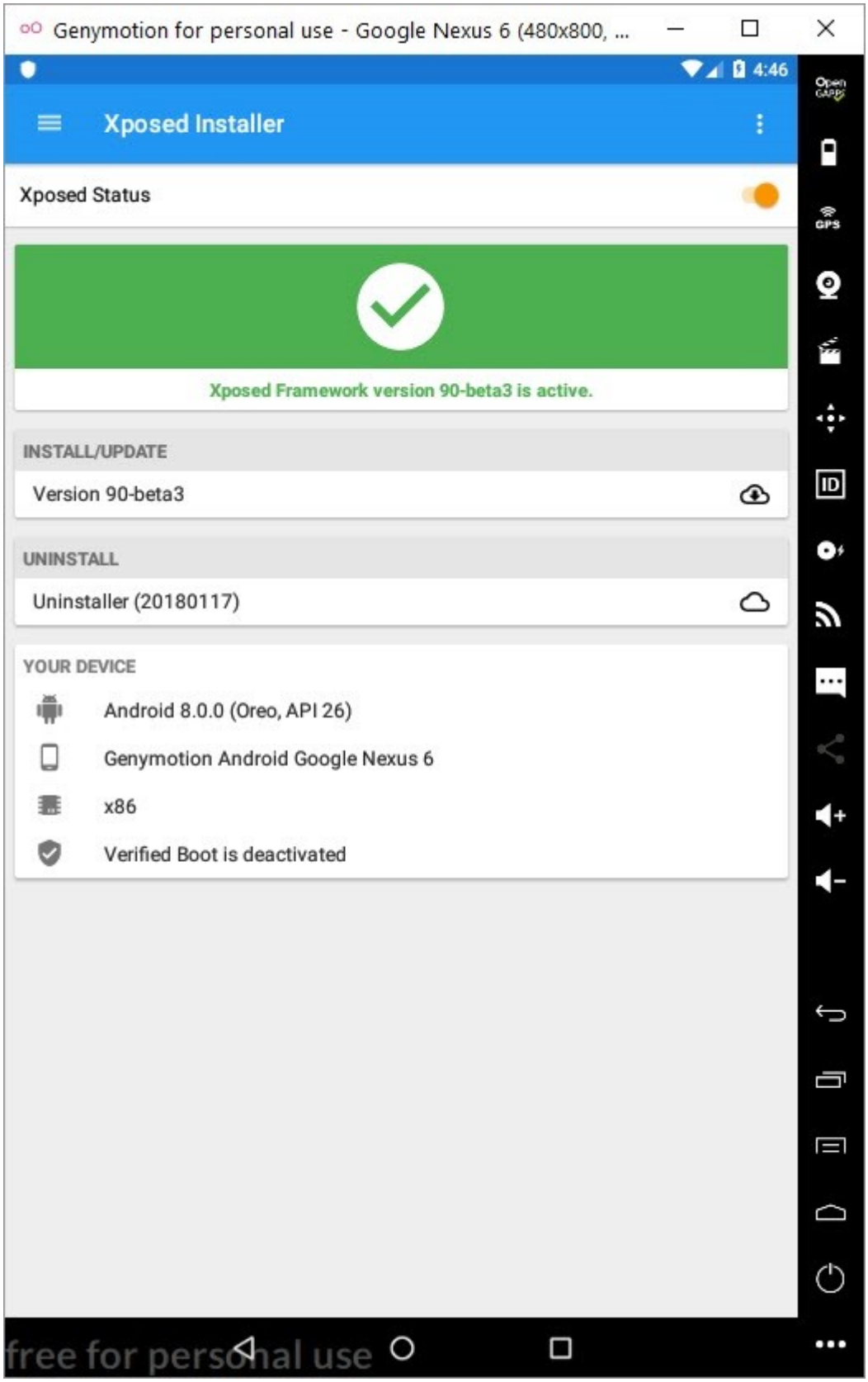


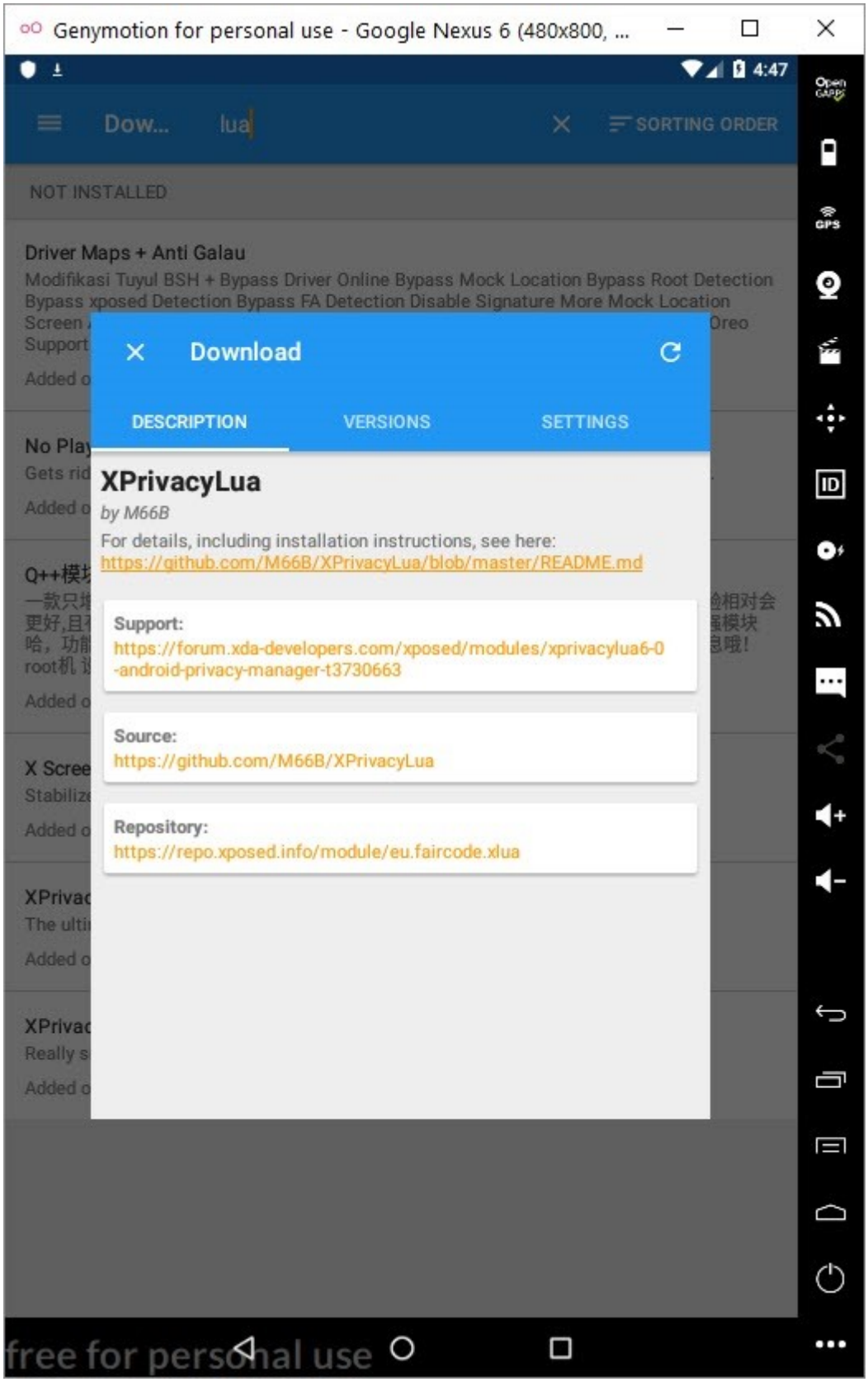


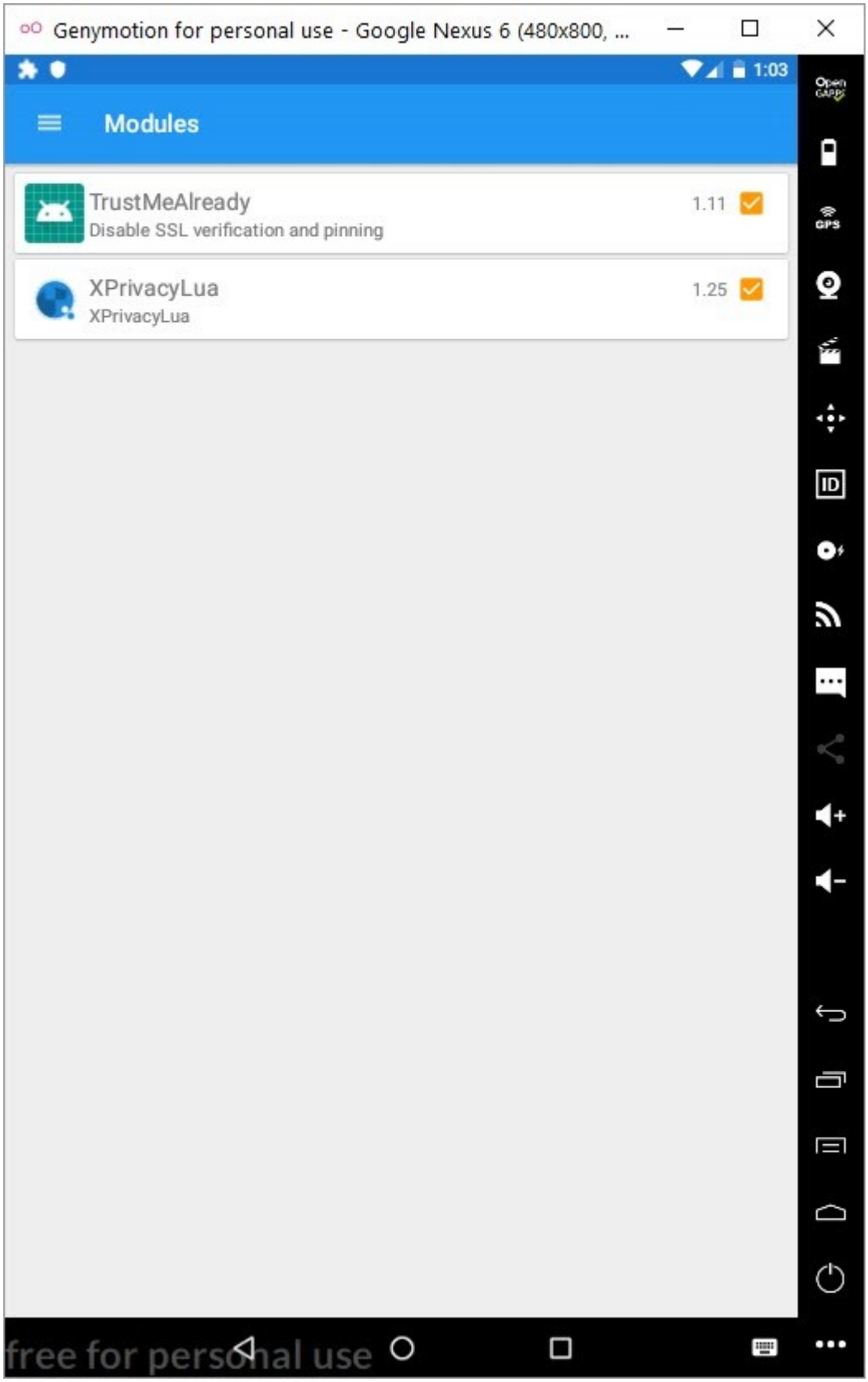
I could have intervened in this function at the source code level, changing the installerPackagename variable to a non-zero value, and then compiling and running it on the Android operating system. But, as Bill Gates said in an interview, “I always hire the laziest person because they find the shortest way to do the job”, I decided to take the lazy way out and look for a shortcut. :)

I’m sure most security researchers who come across such a situation prefer to use the Frida toolkit, but I believe that life should not be limited to Frida, so I decided to look for an alternative tool and a different way. After a short search on Google, I remembered Xposed Framework, which I had used before in penetration tests, especially to bypass SSL Pinning, and has more than 1400 plugins.

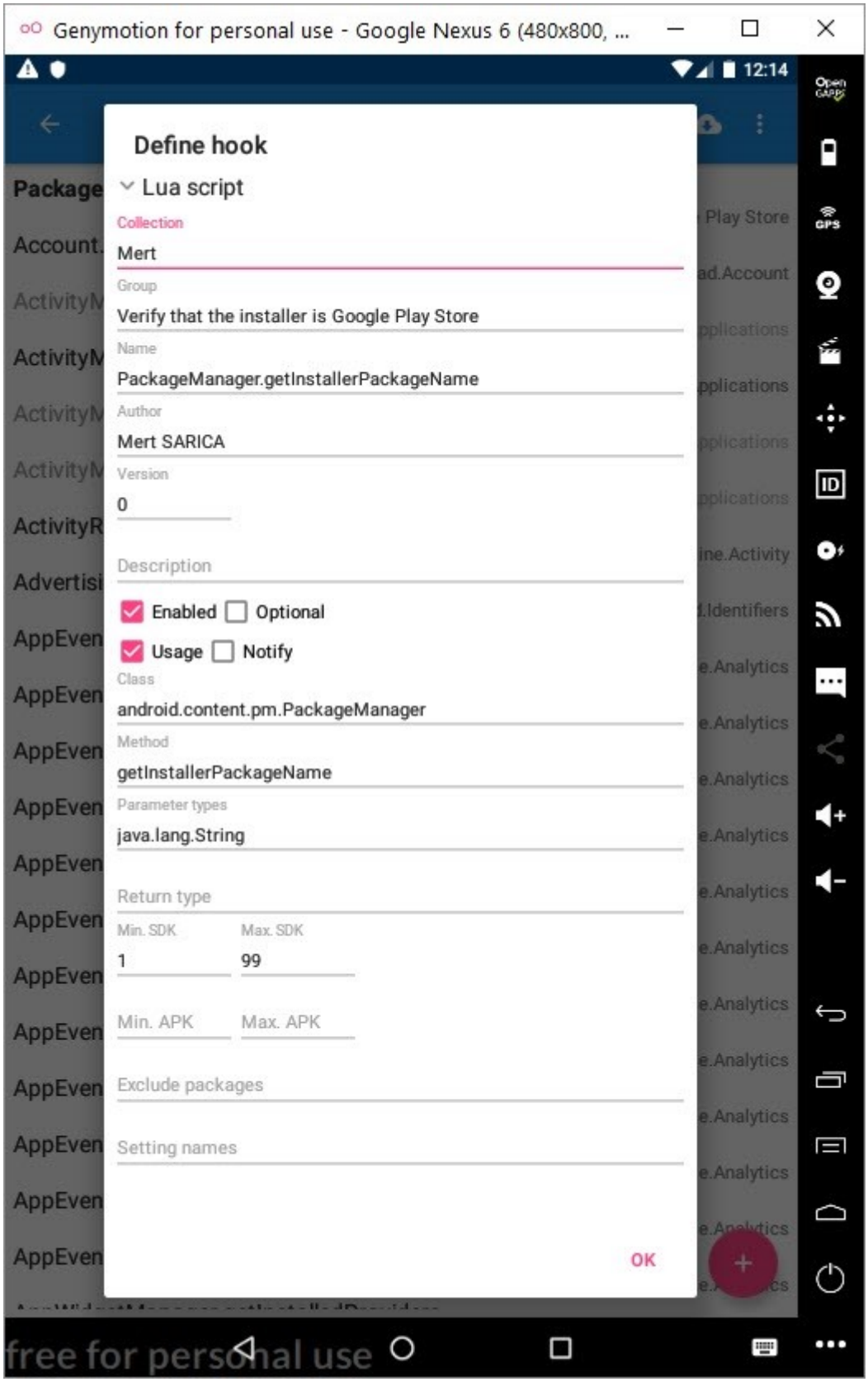
After installing Xposed Framework on the Android Oreo operating system on GenyMotion, I began to look at its plugins. Among the plugins, I was immediately attracted by XPrivaclyLua plugin. As its name suggests, this plugin helps protect your privacy by feeding applications installed on Android with fake information (such as fake location information). It works by roughly hooking onto functions that try to collect this information and providing fake information instead of real information. To shape the plugin to your needs, you need to buy the Pro version and create scripts using Lua programming language.

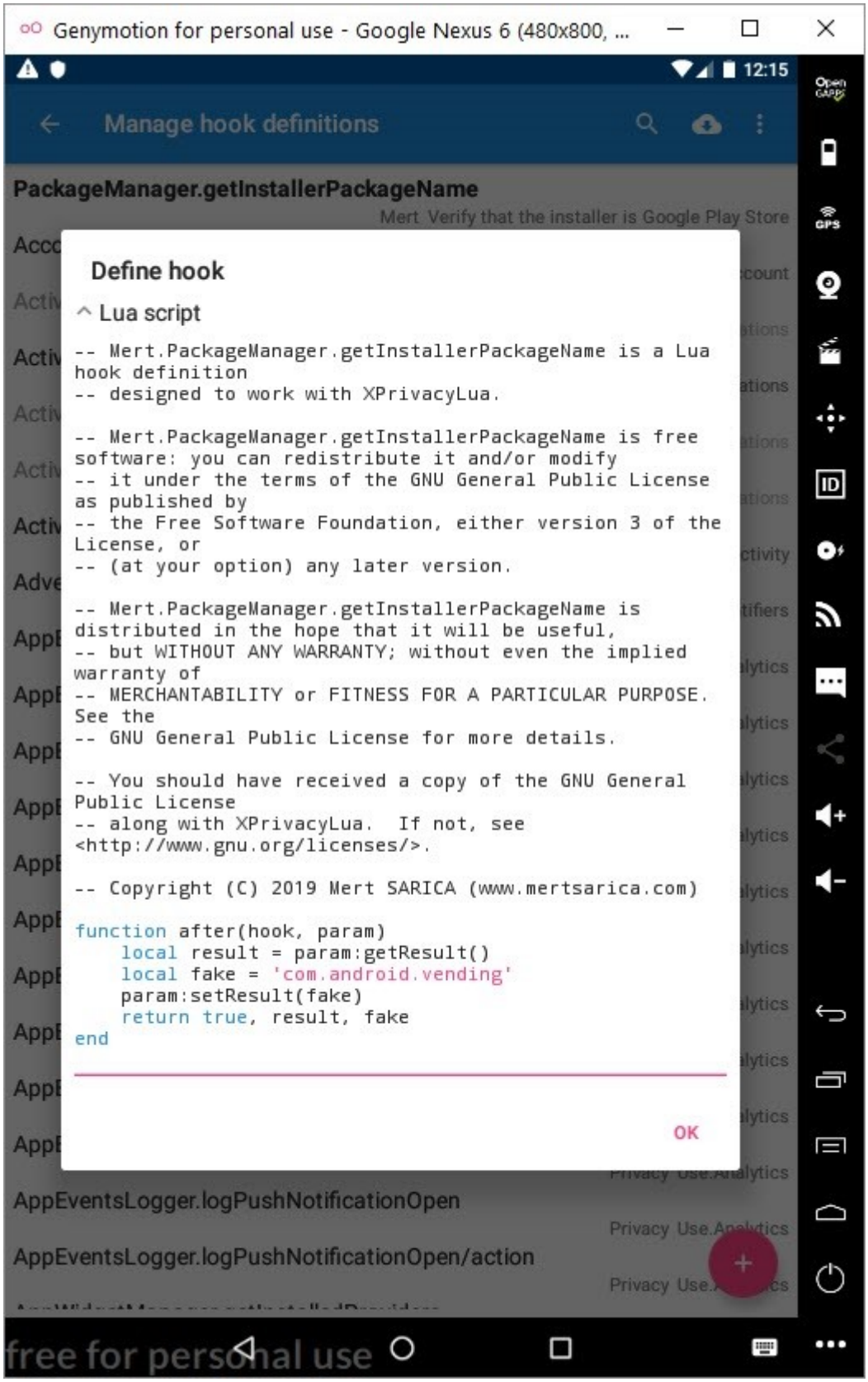

Genymotion for personal use - Google Nexus 6 (480x800, ...
4:46
Xposed Installer
Xposed Status
Xposed Framework version 90-beta3 is active.
INSTALL/UPDATE
Version 90-beta3
UNINSTALL
Uninstaller (20180117)
YOUR DEVICE
Android 8.0.0 (Oreo, API 26)
Genymotion Android Google Nexus 6
x86
Verified Boot is deactivated
free for personal use

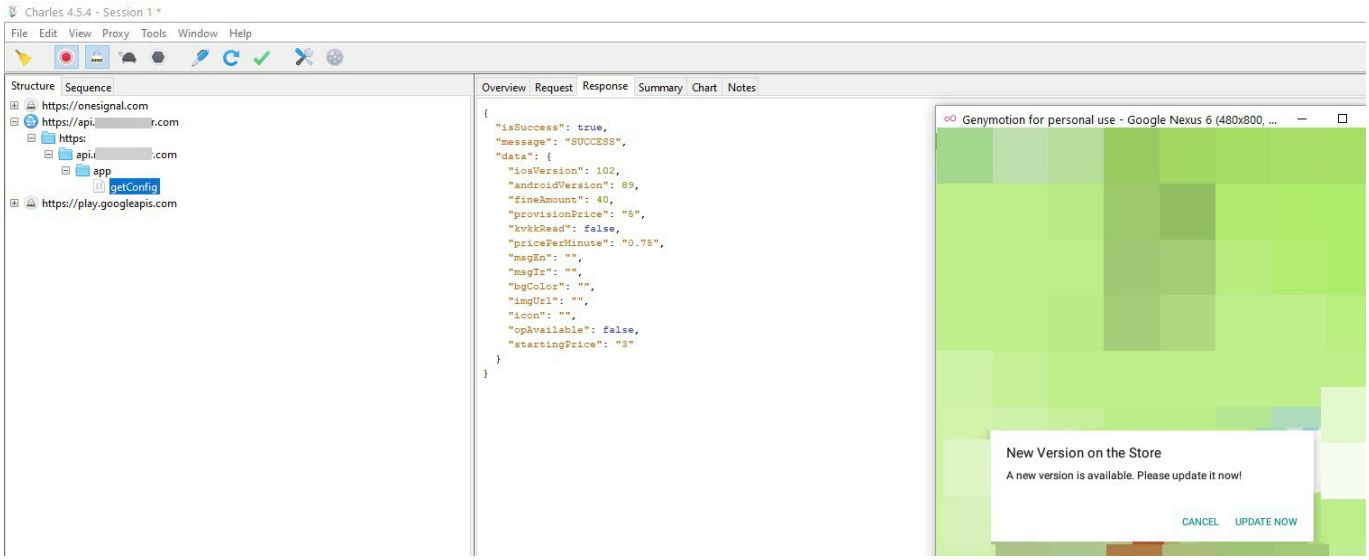

Genymotion for personal use - Google Nexus 6 (480x800, ...
4:47
Dow...
lua
SORTING ORDER
NOT INSTALLED
Driver Maps + Anti Galau
Modifikasi Tuyul BSH + Bypass Driver Online Bypass Mock Location Bypass Root Detection
Bypass xposed Detection Bypass FA Detection Disable Signature More Mock Location
Download
DESCRIPTION
VERSIONS
SETTINGS
XPrivacyLua
by M66B
For details, including installation instructions, see here:
https://github.com/M66B/XPrivacyLua/blob/master/README.md
Support:
https://forum.xda-developers.com/xposed/modules/xprivacylua6-0
-android-privacy-manager-t3730663
Source:
https://github.com/M66B/XPrivacyLua
Repository:
https://repo.xposed.info/module/eu.faircode.xlua
free for personal use



After preparing and activating a small script that changes the installerPackagename variable using Lua, when I ran the application, I was happy to see that the application no longer displayed the previous warning message and I could view the web traffic using Charles Proxy.


Genymotion for personal use - Google Nexus 6 (480x800, ...
12:14
Define hook
Lua script
Collection
Mert
Group
Verify that the installer is Google Play Store
Name
PackageManager.getInstallerPackageName
Author
Mert SARICA
Version
0
Description
Enabled
Optional
Usage
Notify
Class
android.content.pm.PackageManager
Method
getInstallerPackageName
Parameter types
java.lang.String
Return type
Min. SDK
Max. SDK
1
99
Min. APK
Max. APK
Exclude packages
Setting names
OK
free for personal use


Genymotion for personal use - Google Nexus 6 (480x800, ...
12:15
Manage hook definitions
PackageManager.getInstallerPackageName
Mert Verify that the installer is Google Play Store
Define hook
Lua script
-- Mert.PackageManager.getInstallerPackageName is a Lua hook definition
-- designed to work with XPrivacyLua.
-- Mert.PackageManager.getInstallerPackageName is free software: you can redistribute it and/or modify
-- it under the terms of the GNU General Public License as published by
-- the Free Software Foundation, either version 3 of the License, or
-- (at your option) any later version.
-- Mert.PackageManager.getInstallerPackageName is distributed in the hope that it will be useful,
-- but WITHOUT ANY WARRANTY; without even the implied warranty of
-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
-- GNU General Public License for more details.
-- You should have received a copy of the GNU General Public License
-- along with XPrivacyLua. If not, see <http://www.gnu.org/licenses/>.
-- Copyright (C) 2019 Mert SARICA (www.mertsarica.com)
function after(hook, param)
local result = param:getResult()
local fake = 'com.android.vending'
param:setResult(fake)
return true, result, fake
end
OK
AppEventsLogger.logPushNotificationOpen
AppEventsLogger.logPushNotificationOpen/action
free for personal use



I hope this article will provide an alternative tool and method for those who are looking for alternatives to Frida in security research. Hope to see you in the following articles.