Detecting (Absent) App-to-app Authentication on Cross-device Short-distance Channels

Stefano Cristalli, Danilo Bruschi, Long Lu, Andrea Lanzi

December 13, 2019

University of Milan Italy Northeastern University Boston US

Outline

Introduction

Context

- Cross-device communications allow nearby devices to directly communicate bypassing cellular base stations (BSs) or access points (APs) (e.g. **spectral efficiency improvement, energy saving, and delay reduction**, etc.)
- Without the need for infrastructure, **such a technology enables mobile users (e.g., Android) to instantly share information (e.g., pictures and videos)**
- Such technology is also predominat in **IoT environment** where a mobile device is direct connected to the embedded system.

Current Solutions

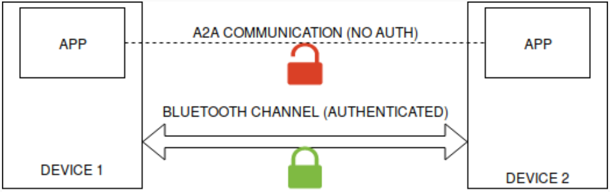
- Several solutions exist for securing cross-device communication. In the Android environment, they allow **authentication of devices and communication channels**.
- Others solutions **restricts apps access to external resources, such as Bluetooth, SMS and NFC**, by defining new SEAndroid types to represent the resources.
- Moroever such **solutions are not able to address several communication channels such as: SMS, Audio, Wi-Fi and NFC** due to of missing important information for the detection purpose.

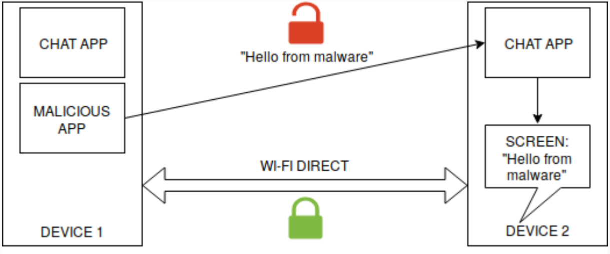
Contributions

- We identify a security problem called **cross-device app-to-app communication hijacking (CATCH)**, which commonly exists in Android apps that use short-distance channels, and afflicts all the tested Android version.
- We provide a solution to the CATCH problem by **designing and developing an authentication scheme detector** that analyzes Android apps to discover potential vulnerabilities
- **Validate the results of our system on Android apps** with manual analysis, and test its resilience in detecting the authentication scheme.

Cross Device Authentication Scheme

Cross-device Authentication Scheme





Threat Model & Attack

- The attacker is able to install a malicious app on the mobile's victim phone.
- The malicious app can therefore craft custom messages to send to the other device, which are displayed as if they were sent from the original app.
- Depending on the particular context, there are some scenarios in which the attack can become very dangerous: **Phishing, Malware delivery, Exploitation**.

Approach Overview

Challenges

- We need to define a **generic scheme that captures the essential logic of app-to-app authentication**.
- We need to define a strategy for differentiating between an if-statement that does not operate on security critical data and an **if-statement that is a part of the authentication scheme**.
- Additionally, the authentication scheme **can be implemented in several ways according to the developer experience**. This adds an additional layer of difficulty for our analysis.

Authentication Scheme Definitions

- We define a **communication** in our model as some exchange of data from A1 to A2, beginning when A2 reads the data from the communication channel.
- We define a **use** of the data as any operation whose result depends on the data itself.
- We define an **authenticated use of the data** as any instruction that needs to be authenticated before access to the data.

Detection Strategy

We define two main check points for our algorithm:

- An **entry point** is an instruction in the code that indicates the start of the communication over the analyzed channel (e.g., data receiving)
- An **exit point** is represented by the first authenticated use of the data coming from the monitored channel.

Detection Algorithm

Algorithm 3.1: Authentication detection

```
input: APK app
output: NO AUTH NEEDED |
        NO AUTH FOUND |
        POSSIBLE AUTH FOUND

  entry_points ← []
  cfg ← computeCFG(app)
  ddg ← computeDDG(app)
 foreach node in cfg
    if isEP(node) then entry_points .add(node)
 end
 if entry_points == [] then return NO AUTH NEEDED

 foreach node in ddg
    if isCondition (node) then
      foreach ep in entry_points
        path ← findPath(ep, node, ddg)
        if path != null
        then
          if isCheckConstant(node, ddg) == false
          then return POSSIBLE AUTH FOUND
        endif
      end
    endif
 end

 return NO AUTH FOUND
```

Technical Details

Technical Details

- Our system is composed of three main components: (1) **Graphs Builder**, (2) **Path Finder** and (3) **App-to-app Authentication Finder**.
- (1) builds an inter-component control flow graph **(ICFG)** and intercomponent data flow graph of the whole app. Finally, the framework builds a data dependency graph (DDG) on top the **IDFG**.
- (2) The Path Finder component traverses the CFG received from Graphs Builder, and **marks entry points for the analyzed channel** based on a predefined list of method signatures.
- (3) App-to-app Authentication Finder applies **further checks** to the paths received from Path Finder, in order **to exclude false positive** results by recognizing checks against constant values.

Experimental Evaluation

Experimental Evaluation

text..... Here we put the experiments that we did

Dataset Composition

text.....

Results

text.....

Case Studies

Data injection on BluetoothChat

text.....

Data injection on Wi-Fi Direct +

text.....

Discussion

Impact & Limitations

text.....

Conclusion & Future works

Conclusion

text.....

Thank you for attention

Questions?