Mobile Containers—The Good, the Bad and the Ugly

Yair Amit
CTO & Co-Founder
Skycure
@YairAmit

Adi Sharabani
CEO & Co-Founder
Skycure
@AdiSharabani
Agenda

- Containers
  - The Good
  - Inherent Limitations
- Android for Work
- App in the Middle
- Demos
- Summary & Recommendations
Sandboxing in Modern OS

Without App Sandbox:
- All user data
- Unrestricted access
- Your app
- All system resources

With App Sandbox:
- Other user data
- No access
- Unrestricted access
- Your sandbox
- Your app
- Other system resources

Source: developer.apple.com
Containers

Another layer of separation
Attempts to restrain the risk
Types of Containers

App-Based Containers
- SDK
- Wrapping
- Enhanced Security (encrypting data in motion & data at rest)
- Separation of Duties (Providing IT policy on business persona)
- Tedious upgrade process
- Narrowed app-selection
- Limitations with iOS
- Susceptible to kernel exploits

OS-Level Containers
- Not available for iOS
- Better user experience
Containers

Breaking Container Security in the Wild
Common Jailbreak Detections

- Existence of directories/files
  - `fileExistsAtPath("/bin/sh")`
  - `fopen("/Applications/Cydia.app","r")`
- Directory permissions
  - `statfs()`
- Process Forking
  - `fork()>=0`
- Cydia scheme detection
  - Check if cydia:// is callable
- Prohibited commands
  - `system()==1`
Malicious Profiles

- Reported to Apple on March 17, 2013

- Problem:
  - Attackers lure victims to install rogue root CAs

- Ramifications:
  - Victim’s traffic analyzed and seamlessly decrypted
  - Some containers are still impacted!

- Solution:
  - Certificate Pinning
  - Hard to be properly implemented in a generic manner
Containers

Breaking Container Separation of Duties in the Wild
Android for Work

Review
Android for Work

- The Premise:
  - **Privacy**: Enterprise IT do not have access to the personal space
  - **Security**: Apps in the personal profile cannot access or manipulate business activity
Android for Work

Where does it break?
While there are two personas, there is only one screen

One screen == employee satisfaction
One screen == hacker satisfaction
Android for Work uses

one screen

with

one Notification system
App in the Middle Flow
Leveraging Notification Features

Malicious app uses the notification service: `BIND_NOTIFICATION_LISTENER_SERVICE`

Malicious app can access business notifications

Malicious app can trigger quick actions on notifications

Malicious app transmits data to a remote server

Emails, meetings, etc – compromised
Reset password link – compromised
Demonstration

Leaking emails is a really big problem.
Android for Work uses one screen with one Accessibility system
Accessibility frameworks are traditionally good source of trouble:

- 2007 – Windows Vista speech recognition exploit
- 2013 – Siri allows to bypass iPhone lock screen
- 2014 – Siri Lets Anyone Bypass Your iPhone's Lockscreen -- Feature or Bug?
- 2015 – iOS 9 allows access to photos and contacts on a passcode locked iPhone
- 2016 – Accessibility clickjacking

Android Accessibility Framework

✓ Has full access to content in other apps (e.g. read emails)
✓ Ability to monitor user activity and take actions accordingly

Accessibility services are global and shared between personas!
App in the Middle Flow
Leveraging Accessibility Features

User enables accessibility features (social engineering, tapjacking)

Malicious app transmits data to a remote server

Business activity captured by the malicious app

Malicious app performs actions on the business persona

Different technology & attack, same App in the Middle concept

Full access (read & write)
No indication to administrator
Two factor authentication – essentially broken
Disclosure

- Reported to Google on December 21, 2016
  - Notification AitM Attack:
    - Google identified the behavior as intended
  - Accessibility AitM Attack:
    - Google identified the behavior as intended
    - Google recommended using this following method:
      ```java
      setPermittedAccessibilityServices(..., List<String> packageNames)
      ```
    - Whitelists approved Accessibility Services
    - We’ve outreached leading EMMs to make sure they implement this API
Caveats

- setPermittedAccessibilityServices(…, List<String> packageNames)

- Sounds good, but...
  - Whitelisting based on a list of Strings is weak
    - Malware can easily name itself as a whitelisted service

- From the documentation:

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true if setting the restriction succeeded. It fail if there is one or more non-system accessibility services enabled, that are not in the list.</td>
</tr>
</tbody>
</table>

Practically, existence of such a service on the device breaks the model
App in the Middle

Getting users to grant permissions
Allow permissions

Social engineering

- Users are led to allow the permission for seemingly good reasons
- Notification API examples:
  - Mirroring notifications
  - Unifying notifications
- Accessibility API examples:
  - Text to speech (for visually impaired users)
  - Translation services
Allow permissions
Accessibility clickjacking

- Introduced by Skycure on RSA 16’
- Victims can be tricked to perform actions without their knowledge or consent
- A tribute to Web-Application security
  - Should actually be called “Accessibility Tapjacking”
- Evolving research (Android 4 → 5 → 6+)
A Few Benign Features

- **Draw Over Apps**
  - Can be presented on top of other apps
    - **SYSTEM_ALERT_WINDOW**
  - Can be used to pass touch events to underlying apps
    - **FLAG_NOT_FOCUSABLE**

- **Accessibility APIs**
... Can Be Dangerous Together

Victims can be tricked to perform actions without their knowledge

https://youtu.be/4cSRq7_Z26s
What About Android 5?

- Original technique was believed to extend till KitKat
- Lollipop introduced an extra protection
  - Tap propagation was not allowed for the “OK” button. A direct tap is required.
- That is not enough...
What about Android 6.0+?

- Draw Over Apps approval has to be done manually.
- Tapjacking is still possible:
  - Full & indirect ramifications of approving the permission are not clear to the user
  - Malware has to lure the victim to approve the DrawOverApps permission
App in the Middle

Summary
App in The Middle

- App in the Middle (AitM) bridges access between remote attacker and sensitive apps
- The two reported issues utilize key Android capabilities to break the secure separation model of Android for Work
  - Security vs. user-experience
  - The framework should provide improved protection against AitM
Apply What You Have Learned Today

- **Next week you should:**
  - Identify who is in charge of your corporate mobile security, and make sure they are aware of threats such as App in the Middle

- **In the first three months following this presentation you should:**
  - Learn more about advanced mobile threats
  - Understand your Mobile Threat Landscape (Malware, Network Threats, Vulnerability Exploitation and Physical access)
  - Gartner, Frost & Sullivan, SANS and others have great papers on the subject

- **Within six months you should:**
  - Select and deploy a Mobile Threat Defense (MTD) solution to proactively protect and gain full visibility into your mobile threats
READ MORE

https://www.skycure.com/blog/app-in-the-middle