OWASP Mobile Top 10 Risk
M4: Unintended Data Leakage

Anant Shrivastava

About Me

- Anant Shrivastava (@anantshri)
- http://www.anantshri.info
- Independent Information Security Consultant
- Focus Area's : Web, Mobile, Linux, Automation
- Current Project:
  - CodeVigilant (codevigilant.com)
    - An initiative to find flaws in open source software and perform a responsible disclosure. Website currently holds 160+ disclosed vulnerability in various wordpress plugins.
  - Android Tamer (androidtamer.com)
    - Live ISO environment for Android Security Researchers. Used by multiple researchers as well as Trainers across the globe.

Agenda

- Understand Data Leakage
- Difference from M2: Insecure data storage
- Example of Unintended data leakage
- How to spot data leakage
- How to prevent it

Data Leakage

- When a developer inadvertently places sensitive information or data in a location on the mobile device that is easily accessible by other apps on the device.
- Typically, these side-effects originate from the underlying mobile device's operating system (OS).
- This will be a very prevalent vulnerability for code produced by a developer that does not have intimate knowledge of how that information can be stored or processed by the underlying OS

M4 v/s M2

- This is what confused most. How does unintended data leakage differ from insecure data storage.

- Simply put
- M2 : Insecure data storage talks about conscious efforts to store data in insecure manner.
- M4: Unintended data leakage talks about OS specific quirks which can cause data leakages.

Common Leakage Points

- URL Caching (Both request and response)
- Keyboard Press Caching
- Copy/Paste buffer Caching
- Application backgrounding
- Logging
- HTML5 data storage
- Browser cookie objects
- Analytics data sent to 3rd parties (ad, social networks etc)

Common Leakage Points

- Disabling screen shots (backgrounding) -- iOS and Android take screen shots of the application before backgrounding the application for improving perceived performance of the application reactivation. However, these screen shots are a cause of security concern due to the potential leak of customer data.
- Key stroke logging -- On iOS and Android, some of the information entered via keyboard is automatically logged in the application directory for use with type-ahead capabilities. This feature could lead to potential leaks of customer data.
- Third-party libraries -- These libraries (such as ad libraries) can leak user information about the user, the device, or the user's location.

Common Leakage Points

- Debugging messages -- Applications can write sensitive data in debugging logs. Setting the logging level to FINE results in log messages being written for all of the data transmitted between the user's device and the server.
- Disable clipboard copy and open-in functionality for sensitive documents displayed as part of the application. MAF currently does not provide the capability to disable copy and open-in functionality and is being targeted for a future release.
- Temporary directories -- They may contain sensitive information.

Example

- Data Leakage via Log's

Example

- Firefox

(CVE-2014-1484) Profile Directory Name Leaks to Android System Log

The random Profile Directory Name is written to the Android System Log (*logcat*) in various locations. For instance, upon Profile creation, the following data is written:

```
1 D/GeckoProfile( 4766): Found profile dir: /data/data/org.mozilla.firefox
2                                           /files/mozilla/24pd90uh.default
```

In Android 4.0 and below, the Android log can easily be read by all apps including malicious ones by acquiring the *READ_LOGS* permission. Android 4.1 has introduced a change to this behavior to prevent such log leakage attacks: The *READ_LOG* permissions are no longer required; however, applications can only listen to their own logs. We will next see how a malicious app can manipulate Firefox for Android to leak its own private log in order to overcome this obstacle.

Preventions

- never log credentials, PII, or other sensitive data to system logs
- remove sensitive data before screenshots are taken
- disable keystroke logging per field, and utilize anti-caching directives for web content
- debug apps before releasing them to observe files created
- review third party libraries introduced and the data they consume, and
- test applications across as many platform versions as possible.

References

- https://www.owasp.org/index.php/Mobile_Top_10_2014-M4
- http://securityintelligence.com/vulnerabilities-firefox-android-overtaking-firefox-profiles/
- http://docs.oracle.com/middleware/mobile200/mobile/develop-oepe/oepe-maf-secure-dev-pract.htm
- https://www.owasp.org/index.php/IOS_Developer_Cheat_Sheet

Question Time