Risks behind Device Information Permissions

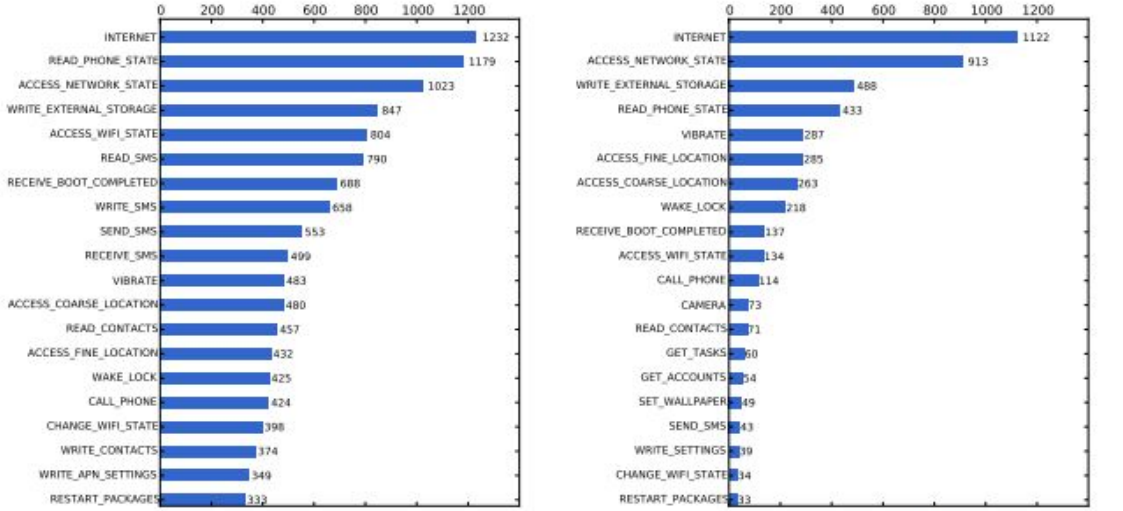
Anthony Hewins and Maria McCulley

Outline:

- Related Works
- smsStealer attack automation and roadblocks
- Our Proposed Solutions

Related Works:

[1]



(a) Top 20 Permissions Requested By 1260 Malware Samples

(b) Top 20 Permissions Requested by 1260 Top Free (Benign) Apps on the Offical Android Market

Figure 5. The Comparison of Top 20 Requested Permissions by Malicious and Benign Apps

Related Works:

- Top 1,865 free applications evenly distributed among the twenty-two categories on the Google Play store [2]
  - 167 have access to device identifiers (≈ 9%)
  - 114 stream this data immediately upon receiving it (≈ 68% of those who have access)
  - Conclusion: This private information is constantly leaving the device
- TaintDroid analysis of 30 popular applications [3]
  - 21 had both READ_PHONE_STATE and INTERNET permissions
  - 9 transmitted IMEI (30% of total applications)
  - 7 either did not have EULA or did not reveal they were collecting IMEI
  - Conclusion: Users have no way of knowing how their information is being used

Related Works

- AndroidLeaks investigated 25,976 free applications from thirteen Android markets [4]
  - 7,414 were found to potential privacy leaks
  - Phone leaks (leakage of IMEI or IMSI) compromised most of these

Table 1. Breakdown of Leaks by Type

Leak Type	# Leaks	% of all Leaks	# apps with leak	% apps with leak
Phone	53,281	92.99%	6912	28.39%
Location	3,405	5.94%	969	3.98%
WiFi	266	0.46%	79	0.32%
Record Audio	347	0.61%	115	0.47%

  - Conclusion: Out of all of our data, device identifiers are the most frequently leaked

Related Works:

- 1,100 popular free applications [5]
  - 246 apps attempted to obtain device identifiers
  - 216 apps attempted to obtain IMEI
  - 60% of all calls were to retrieve IMEI
- Conclusion: IMEI is the most popular device identifier

Table 2: Access of Phone Identifier APIs

Identifier	**# Calls**	**# Apps**	**# w/ Permission***
Phone Number	167	129	105
IMEI	378	216	184†
IMSI	38	30	27
ICC-ID	33	21	21
Total Unique	-	246	210†

* Defined as having the `READ_PHONE_STATE` permission.

† Only 1 app did not also have the `INTERNET` permission.

Related Works:

- [5] came up with the following conclusions about the leakage of device Identifiers
  - Device identifier are frequently sent in plaintext
  - Phone identifiers are used as device fingerprints to track users and tie their device to other personal identifiable information (PII)
  - This information can then be sold to advertisement and analytic servers
- Comparison to our work:
  - We provide a much more complete picture of the dangers of device information

smsStealer: Using Selenium WebDriver

- Selenium operates a web browser via Java code
- When resetting user passwords, this makes it really fast, users likely won't react in time
- You don't even need a human to be present, you could just fill a database up with everything you get
- Can also automatically obtain additional information from the user: zip code, family members, sometimes where they live, etc.

Just one problem...

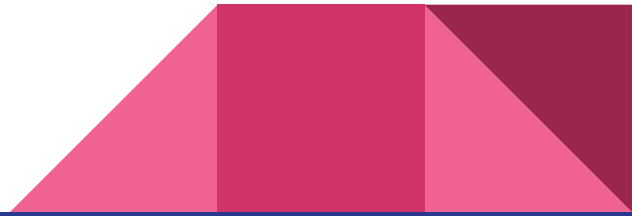
- Selenium updated itself and now Selenium doesn’t seem to be working on anything
- Firefox also updated itself so previous versions of selenium aren’t compatible
- HTMLUnit (a headless browser) which was another option, also doesn't work on selenium
- Either going to propose this as a possibility instead or use another software type (AutoHotKey seems to be a far worse alternative, but it works)

Solutions:

- Break down READ_PHONE_STATE into phone status and phone identity
  - Phone status - hasCarrierPrivileges(), getCallState(), etc.
  - Phone Identity - getDeviceId(), getLine1Number, getSimSerialNumber(), etc.
  - Why?
    - Phone status is necessary for basic functionality of many applications
    - Phone identity is dangerous and is only needed by the default messaging/phone application
      - And at the same time, they often make no sense for an app to have, especially IMEI/MEID numbers
    - Current solutions are not enough
      - Marshmallow
      - Anonymity tools such as IdentiDroid [6]

smsStealer Solutions in brief (confirmed)

1. Verification messages are *in-app*, not by text
    - By not using a Broadcast of an SMS message, there's no way to get verification messages
2. Use email instead (which simply means users really have to guard their emails well, which is already the norm)



Week 1

-Begun Android research

Week 2

-Begun READ_PHONE_STATE Permission research
-Conducted exploit analysis

Week 3

-Expanded research to include other permissions
-Conducted exploit analysis

Week 4

-Searched for statistics/research to back up claims
-Begun SMS proof of concept

Week 5

-Prepared for Midterm Presentation
-Designed experiments to be run on tablet

Week 6

-Began application experimenting
-Continued proof of concept for SMS attack

Week 7

-Continued drafting official first draft of paper
-Analyzed malware for device identifier misuse

Week 8

-Analyze areas of weakness in our paper and revise
-Begin poster

Week 9

-Prepare for Mid-SURE
-Complete paper and poster
-Finish paper

Week 10

References

[1] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution", *2012 IEEE Symposium on Security and Privacy*, 2012.

[2] L. Batyuk, M. Herpich, S. Camtepe, K. Raddatz, A. Schmidt and S. Albayrak, "Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications", *2011 6th International Conference on Malicious and Unwanted Software*, 2011.

[3] W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel and A. Sheth, "TaintDroid", *Communications of the ACM*, vol. 57, no. 3, pp. 99-106, 2014.

[4] C. Gibler, J. Crussell, J. Erickson and H. Chen, "AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale", *Trust and Trustworthy Computing*, pp. 291-307, 2012.

[5] W. Enck, D. Octeau, P. McDaniel and S. Chaudhuri, "A study of android application security", *Proceedings of the 20th USEN*

[6] B. Shebaro, O. Oluwatimi, D. Midi and E. Bertino, "IdentiDroid: Android can finally Wear its Anonymous Suit", *Transactions on Data Privacy*, vol. 7, no. 1, pp. 27-50, 2014.*IX conference on Security*, pp. 21-21, 2011.

That’s all

Questions?

http://reu16.weebly.com/