Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android

Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard
IAIK, Graz University of Technology, Austria

WiSec 2016, Darmstadt, Germany, 18th July 2016
Contributions

Side-channel attack to infer browsing behavior

- Unprivileged application
- Data-usage statistics
- High accuracy
- Also works when traffic is routed through Tor
- READ_HISTORY_BOOKMARKS does not provide protection
Website Fingerprinting

Traditional attack scenario

- Attacker located somewhere on the victim’s network
- Traffic analysis techniques to infer browsing behavior
Website Fingerprinting

Attack scenario against smartphones

- Malicious application running in unprivileged mode
- Observe information “leaking” from browser application
Data-Usage Statistics

What is this?
- Track the **amount of incoming/outgoing network traffic**
- Users can stick to their data plan
- Available to all apps w/o any permission

Availability
- `/proc/uid_stat/[uid]/tcp_rcv|tcp_snd`
- Android API `TrafficStats.getUidRxBytes, .getUidTxBytes`
- How to get `uid`?
  - `ActivityManager.getRunningAppProcesses()` (REAL_GET_TASKS?)
  - `PackageManager.getInstalledApplications()`

*High resolution* (single TCP packet lengths)
Data-Usage Statistics

Experiment

- Local server hosting a website (tcpdump)
- Launch website on Android (data-usage statistics)
Usage Statistics for Real Websites

Websites are distinguishable

- **Stable**: signatures of repeated visits to the same page are similar
- **Diverse**: signatures of different pages vary
Adversary Model and Attack Scenario

Adversary model

- Traditional: nw-based attacker
- Unprivileged app distributed via app market

Attack

1. Training phase (offline)
2. Attack phase (online)
Website Fingerprinting

Training phase

- Observe data-usage statistics while loading specific websites
- ⇒ build signature database
- No “fancy” machine-learning approach
- ⇒ no expensive training phase necessary
Website Fingerprinting

Attack phase

1. Distribute malicious application
2. Observe data-usage statistics for browser application
3. Infer visited website by means of signature database
   - Similarity metric for traces
     \[
     \text{SIM}(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|}
     \]
Results
Intra-day classification rate
- 100 most popular websites globally

89% of 500 page visits
confusion of google*.* pages

98% of 500 page visits
with google*.* pages merged
Results

Inter-day classification for Tor

- 100 most popular websites in the US
### Results

Websites with the highest number of misclassifications

<table>
<thead>
<tr>
<th>∆</th>
<th>Website</th>
<th># misclassifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 days</td>
<td>ask.com</td>
<td>5 times</td>
</tr>
<tr>
<td>2 days</td>
<td>twitch.tv</td>
<td>5 times</td>
</tr>
<tr>
<td>2 days</td>
<td>cnn.com</td>
<td>3 times</td>
</tr>
<tr>
<td>5 days</td>
<td>bbc.com</td>
<td>5 times</td>
</tr>
<tr>
<td>5 days</td>
<td>indeed.com</td>
<td>5 times</td>
</tr>
<tr>
<td>5 days</td>
<td>nytimes.com</td>
<td>5 times</td>
</tr>
<tr>
<td>5 days</td>
<td>twitch.tv</td>
<td>5 times</td>
</tr>
<tr>
<td>5 days</td>
<td>espn.go.com</td>
<td>4 times</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Work</th>
<th>Exploited information</th>
<th>Countermeasure</th>
<th># websites</th>
<th>Classification rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>Client-side data-usage statistics</td>
<td>None</td>
<td>500</td>
<td>97%</td>
</tr>
<tr>
<td>Jana and Shmatikov [JS12]</td>
<td>Client-side memory footprint</td>
<td>None</td>
<td>100</td>
<td>35%</td>
</tr>
<tr>
<td>Ours</td>
<td>Client-side data-usage statistics</td>
<td>Tor</td>
<td>100</td>
<td>95%</td>
</tr>
<tr>
<td>Wang et al. [WCN+14]</td>
<td>TCP packets</td>
<td>Tor</td>
<td>100</td>
<td>95%</td>
</tr>
<tr>
<td>Wang and Goldberg [WG13]</td>
<td>TCP packets</td>
<td>Tor</td>
<td>100</td>
<td>91%</td>
</tr>
<tr>
<td>Cai et al. [CZJJ12]</td>
<td>TCP packets captured via tshark</td>
<td>Tor</td>
<td>100</td>
<td>84%</td>
</tr>
<tr>
<td>Panchenko et al. [PNZE11]</td>
<td>Client-side tcpdump</td>
<td>Tor</td>
<td>775</td>
<td>55%</td>
</tr>
<tr>
<td>Herrmann et al. [HWF09]</td>
<td>Client-side tcpdump</td>
<td>Tor</td>
<td>775</td>
<td>3%</td>
</tr>
</tbody>
</table>

## Advantages

- Ease of applicability *(unprivileged app vs on the wire)*
- Computational performance *(no training vs 608 000 CPU seconds)*
- Classification rates
- No traffic noise due to other apps
Countermeasures
Against NW-based fingerprinting attacks

- Traffic morphing, HTTPOS, BuFLO, Glove
- Tor?

Client-side countermeasures

- Permission-based approaches? [ZDH+13]
  - Request permission to monitor data-usage statistics?
  - Let developers specify how statistics should be published?

⇒ update data-usage statistics on a more coarse-grained level
Conclusions

Fundamental weaknesses in Android

- Seemingly innocuous information
- ...that turns out to be a serious information leak

Unprivileged app can infer browsing behavior, although

- Orweb or “private/incognito” modes do not store browsing history
- Traffic is routed through Tor
- READ_HISTORY_BOOKMARKS should protect this sensitive information

⇒ Privacy issue
Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android

Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard
IAIK, Graz University of Technology, Austria

WiSec 2016, Darmstadt, Germany, 18th July 2016
Bibliography I

Touching from a Distance: Website Fingerprinting Attacks and Defenses. 

[HWF09] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 
Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. 

Memento: Learning Secrets from Process Footprints. 

Website Fingerprinting in Onion Routing Based Anonymization Networks. 

Effective Attacks and Provable Defenses for Website Fingerprinting.


Improved Website Fingerprinting on Tor.


Identity, Location, Disease and More: Inferring Your Secrets from Android Public Resources.