Website Fingerprinting Defenses at the Application Layer

Giovanni Cherubin\textsuperscript{1} Jamie Hayes\textsuperscript{2} Marc Juarez\textsuperscript{3}

\textsuperscript{1}Royal Holloway University of London
\textsuperscript{2}University College London
\textsuperscript{3}imec-COSIC KU Leuven

19th July 2017, PETS’17, Minneapolis, MN, USA
Introduction: Website Fingerprinting (WF)
Tor Hidden Services (HS)

- HS: user visits xyz.onion without resolving it to an IP
- Examples: SecureDrop, Silkroad, DuckDuckGo, Facebook
Website Fingerprinting on Hidden Services (HSes)

• WF adversary can distinguish HSes from regular sites

• Website Fingerprinting in HSes is more threatening:
  - **Smaller world** makes HSes more identifiable
  - HS users vulnerable because content is *sensitive*
Website Fingerprinting defenses

**WF Defenses**
- BuFLO
- Tamaraw
- CS-BuFLO
- WTF-PAD
- ...

These are TCP packets or Tor messages

User → Entry → Middle

- **Dummy**
- **Real**
Application-layer Defenses

- Existing defenses are designed at the network layer

Key observation: identifying info originates at app layer!

Latent features: $F_1, \ldots, F_n$

Identifying info: $O_1, \ldots, O_n$

Web content

Last layer of encryption

Adversary
Pros and Cons of app-layer Defenses

The main advantage is that they are easier to implement:

• do not depend on Tor to be implemented

Cons:

• padding runs end-to-end
• may require server collaboration:

...but HSes have incentives!
<table>
<thead>
<tr>
<th>LLaMA</th>
<th>ALPaCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Client-side (FF add-on)</td>
<td>• Server-side (first one)</td>
</tr>
<tr>
<td>• Applied on HTTP requests</td>
<td>• Applied on hosted content</td>
</tr>
<tr>
<td>• More latency overhead</td>
<td>• More bandwidth overhead</td>
</tr>
</tbody>
</table>

(two different solutions, **not** a client-server solution)
ALPaCA

- Abstract web pages as **num objects** and **object sizes**: pad them to match a target page.

- Does not impact user experience:
  e.g., comments in HTML/JS, images’ metadata, *hidden* styles.
Example: protect a SecureDrop page

- Strategy 1: target page is Facebook
ALPaCA strategies (2)

- Strategy 2: pad to an “anonymity set” target page

Defines num objects and object sizes by:

- Deterministic: next multiple of $\lambda$, $\delta$
- Probabilistic: sampled from empirical distribution
LLaMA

- Inspired by Randomized Pipelining
  Goal: randomize HTTP requests

- Same goal from a FF add-on:
  - Random delays ($\delta$)
  - Repeat previous requests ($C_1$)
Evaluation: methodology

- Collect **with** and **without** defense: 100 HSes (cached)
  - Security: *accuracy* of attacks
    - \(k\text{NN}, k\text{-Fingerprinting (kFP)}, CUMUL\)
  - Performance: overheads
    - *latency* (extra delay)
    - *bandwidth* (extra padding/time)
ALPaCA: results

- From 60% to 40% decrease in accuracy
- 50% latency and 85% bandwidth overheads
LLaMA: results

- Accuracy drops between 20% and 30%
- Less than 10% latency and bandwidth overheads
Take aways

- WF defenses at the app layer are **easier to implement**

- **HSes have incentives** to support server-side defenses:
  
  *SecureDrop* has implemented a prototype of ALPaCA

- ALPaCA is running on a HS: [3tmaadslguc72xc2.onion](3tmaadslguc72xc2.onion)

- Source code: [github.com/camelids](github.com/camelids)
Website Fingerprinting: deployment
Closed vs Open World

Closed world

Open world
The HS world

- Exploratory crawl: 5,000 HSes (from Ahmia.fi)
- Stats for the HS world (from intercepted HTTP headers)
  - Distribution of types, sizes and number of resources
    - Most HSes are small compared to an average website
- Few HSes have any JS or 3rd-party content
  - JS: less than 13% ⇒ Assumption: no JS
  - 3rd party content: less than 20% ⇒ Assumption: no 3rd parties
ALPaCA (2)

- Two ways to generate a target page from the original:
  - **Deterministic** (D-ALPaCA): takes params $\lambda$, $\delta$
    - Number of objects in the page is the next multiple of $\lambda$
    - Sizes of objects are next multiple of $\delta$
  - **Probabilistic** (P-ALPaCA): sample number of objects and object sizes from empirical distributions
Limitations and Future Work

- ALPaCA can only make sites bigger, but not smaller
- What’s the optimal padding at the app layer? Lack of a thorough feature analysis.
- How do the distributions change over time? How do we update our defenses accordingly?
  - How does the strategy need be adapted as HSes adopt our defense(s)?
Results

- ALPaCA: comparable to previous defenses
  - 60 to 40% decrease in accuracy
  - 50% latency and 80% bandwidth overheads
- LLaMA: lightweight but low protection
  - 20 to 30% decrease in accuracy
  - Less than 10% latency and bandwidth overheads