Var-CNN and DynaFlow: Improved Attacks and Defenses for Website Fingerprinting

Sanjit Bhat (PRIMES)  David Lu (PRIMES)  Albert Kwon (MIT)  Srini Devadas (MIT)
sanjit.bhat@gmail.com  davidboxboro@gmail.com

May 20th, 2018
PRIMES Conference
Motivation and Background
Anonymity matters

- Whistleblowers
- Governmental suppression of political opinion
- Censorship circumvention


http://facecrooks.com/Internet-Safety-Privacy/To-be-anonymous-or-not-to-be-should-you-use-your-real-name-on-the-Internet.html/

http://www.dmnews.com/social-media/what-if-people-want-their-internet-anonymity-back/article/338654/
The internet provides limited anonymity

Sender (Alice)  
From: Alice  
To: Bob  

Adversary

Receiver (Bob)  
From: Alice  
To: Bob
A supposed fix - Tor: The Onion Router

- Alice connects to the Tor network
A supposed fix - Tor: The Onion Router

- Alice obtains a list of Tor nodes from the Tor network
A supposed fix - Tor: The Onion Router

- Alice chooses 3 Tor nodes to make a connection to Bob
- No Tor nodes know the identities of both Bob and Alice
Traffic analysis attacks

- Adversary correlates Alice and Bob’s traffic
- Only works when adversary intercepts both entry and exit points
Website fingerprinting (WF) attacks

- Adversary collects database offline and uses it to fingerprint online
- Only needs 1 link in the chain - weaker threat model
Simplified WF attack scenario

- Each website exhibits characteristic load behavior
Var-CNN: Automated feature extraction using variations on CNNs
Why automated feature extraction?

- Uses raw Tor traffic sequences: incoming/outgoing direction, timestep
- Resists network protocol changes
- Could discover more optimal features than humans can find
Dilated convolutions

- Packet sequence inherently time-dependent

Dilated convolutions

- Sacrifice fine-grain detail for broader field of view

Other techniques

- Cumulative features
  - Total number of packets
  - Number of incoming and outgoing
  - Ratio of incoming to total and outgoing to total
  - Total transmission time
  - Average number of packets per second

- Confidence thresholds
  - Threshold for attacker certainty
  - Adjust types of classification made

```
S1: 0.5  S2: 0.4  UM: 0.1
```

Conf = 0.7

Normal Output

Softmax Layer
Ensemble model

- Using timesteps should leak more info to attacker
- No past pre-extracted timing features performed well
Var-CNN Results
Experimental setup

- Wang et al. k-NN data set - blocked pages for monitored, popular pages for unmonitored
- ≤ training data used by competing attacks
- Re-randomize train/test sets and average results over 10 trials
- Metrics
  - *True Positive Rate* (TPR) - Prop. of monitored sites correctly classified
  - *False Positive Rate* (FPR) - Prop. of unmonitored sites incorrectly classified
Ensemble model and confidence threshold

- Alone, time model is worse than direction model
- However, their performance is additive
- TPR and FPR decrease as confidence threshold increases
Open-world performance

- 5% better TPR than SDAE
- Over a sixth the FPR of SDAE
- 3% better TPR than $k$-FP
- Nearly half the FPR of $k$-FP
DynaFlow: A new defense based on dynamically-adjusting flows
Existing WF defenses

1) **Limited defenses** - Designed to counter existing attacks
   **Drawback:** No provable guarantees

2) **Supersequence-based defenses** - Sends “Supersequence” of web trace
   **Drawbacks:** Requires constantly updated database; does not protect static content

3) **Constant-flow defenses** - Sends a continuous stream of network traffic
   **Drawback:** High overheads
# Advantages of DynaFlow

<table>
<thead>
<tr>
<th></th>
<th>Low Latency</th>
<th>Low Bandwidth Usage</th>
<th>Strong Security Guarantees</th>
<th>Protects Dynamic Content</th>
<th>No Database Required</th>
<th>Highly Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynaFlow</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BuFLO [13]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Tamaraw [7]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Supersequence [40]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Walkie-Talkie [42]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Glove [29]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>WTF-PAD [21]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Decoy Pages [32]</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>LLaMA [10]</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
Overview of DynaFlow

**Our goal:** to construct a defense with similar guarantees as prior art but with significantly lowered overheads.

**Three Components:**
1) Burst-pattern morphing
2) Constant traffic flow with dynamically changing intervals
3) Padding the number of bursts
**Burst-pattern morphing**

- Traffic is morphed into fixed **bursts**: 1 outgoing packet followed by 4 incoming packets
- Dummy packets added to morph traffic

**Before padding:**

```
Out In Out In In In
```

**After padding (red packets are dummy packets):**

```
Out In In In In Out In In In In
```
Inter-packet timing

- Packets are sent every $t$ seconds
- The value of $t$ dynamically changes to fit the loading page
- There are three tunable parameters: $a$, $b$, $T$
  - The value of $t$ changes every $b$ bursts
  - Up to $a$ adjustments total
  - The value of $t$ is chosen from the set $T = \{t_1, \ldots, t_k\}$
The number of bursts

● The number of bursts is padded to $\{[m], [m^2], [m^3], \ldots \}$

● Advantages of padding to a power of $m$
  ○ Significantly mitigate privacy loss
  ○ Incur reasonably-small overhead

● Example: when $m = 2$, the bandwidth overhead is under 100%
DynaFlow Results
Open-world eval. against existing attacks

DynaFlow against existing attacks. All values are in %.

<table>
<thead>
<tr>
<th></th>
<th><em>k</em>-NN [40]</th>
<th></th>
<th><em>k</em>-FP [14]</th>
<th></th>
<th>Var-CNN</th>
<th>TOH</th>
<th>BWOH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TPR</td>
<td>FPR</td>
<td>TPR</td>
<td>FPR</td>
<td>TPR</td>
<td>FPR</td>
<td></td>
</tr>
<tr>
<td>No defense:</td>
<td>84.5</td>
<td>2.5</td>
<td>86.3</td>
<td>1.6</td>
<td>89.1</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>Medium security:</td>
<td>15.4</td>
<td>20.6</td>
<td>5.0</td>
<td>1.6</td>
<td>10.8</td>
<td>3.0</td>
<td>23</td>
</tr>
<tr>
<td>High security:</td>
<td>5.9</td>
<td>69.0</td>
<td>4.4</td>
<td>40.1</td>
<td>0.6</td>
<td>0.9</td>
<td>28</td>
</tr>
</tbody>
</table>
Open-world evaluation against prior art

- 31% F1 score: 29% TPR, 11% FPR
  - DynaFlow: 101% overhead (29% TOH, 73% BWOH)
  - Prior art: 138% overhead (40% TOH, 98% BWOH)
- Gap increases for larger F1 scores
**Conclusion**

- **Var-CNN** uses novel variants of CNNs to improve upon prior work:
  - Be highly tunable in terms of TPR-FPR trade-off
  - Outperform all prior attacks, all while using ≤ amount of training data
- **DynaFlow** overcomes challenges of prior WF defenses:
  - Lower overhead than prior work while providing stronger security
  - Protects dynamic content & no database required
- **Current status**
  - Preprint on arXiv
  - All code and data sets publically available
Acknowledgements

Thank you to:

● Our parents
● Albert Kwon, for providing advice every step of the way
● Prof. Devadas, for giving feedback on the paper and running PRIMES CS
● Dr. Gerovitch and the PRIMES program, for providing research opportunities to high school students and sponsoring AWS bills and a GPU :-)}
Appendix of Slides
Var-CNN architecture

- VGG-16 Convolutional Neural Network (CNN) - ImageNet competition
- Multiple blocks composed of multiple layers for deeper feature extraction
Scaling performance - FPR

- FPR is incredibly important as open-world size increases
- Training on greater numbers of unmonitored sites retains TPR while reducing FPR
- Var-CNN scales better to larger open-worlds than prior-art attacks
Scaling performance - runtime

- Runtime scales linearly, better than prior models
The optimal attacker

Overview:
- Knows the exact probability that a website $w$ is visited, generating defended trace $t$
- Uses this information to make the best guess for which website $w$ is visited when he sees a trace $t$
- We can use this information to calculate what the optimal attacker would guess.

Measuring accuracy:
- **F1-score** — harmonic mean of precision and recall (TPR)
Future work

- More powerful deep learning models for Var-CNN
  - Computer vision architectures - DenseNet
  - Recurrent Neural Network architectures - LSTM with Synthetic Gradients
- Find a better way to determine optimal DynaFlow parameters
  - Currently, we sweep parameters one at a time
- Further reduce DynaFlow overheads
  - Total overhead sum can still exceed 100% for stronger configurations