Adversarial AI to Prevent Microarchitectural Website Detection Attacks

Sddec21-13: Ege Demir, Thane Stoley, Aaron Anderson, Sean McClannahan
Adviser and Client: Berk Gülmezoğlu

Microarchitectural Attacks

- Exploit of the Last Level Cache (LLC) that can be abused by malicious people to gather your data.
- With advancements of AI technologies, current browser defenses are not enough to counter the problem.

An example of cache usage across 3 different websites, from previous research into this topic.

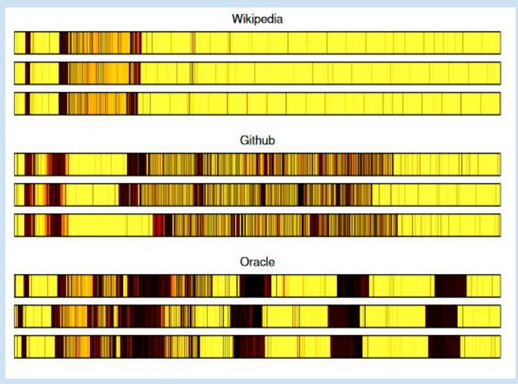
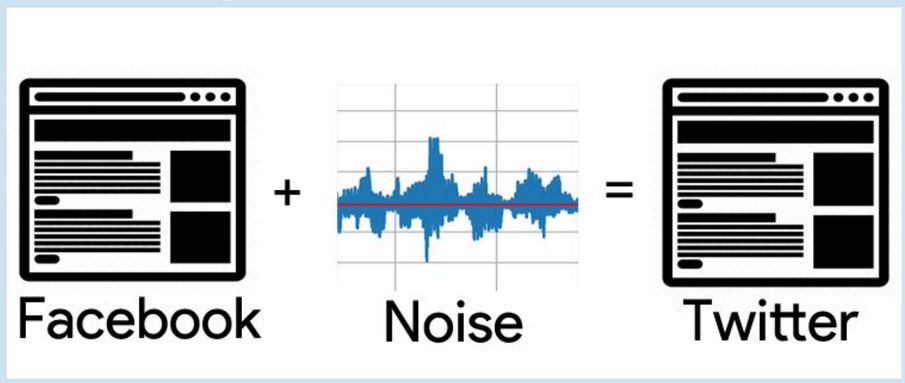


Figure: Robust Website Fingerprinting Through the Cache Occupancy Channel (Shusterman et al.)

Adversarial AI techniques

- They're techniques used to trick neural networks into misclassifying results.
- The most common example of its usage being with images, using "noise", filling the image or cache with dummy data.
- It can confuse an attacking code by making it think the image or website is one thing, when it's really another.



Functional Requirements

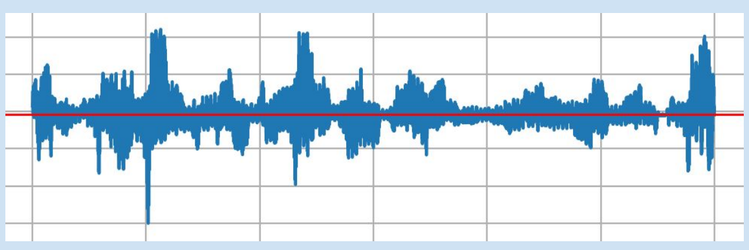
- Javascript-based cache monitoring code
- Python-based AI model that classifies websites based on output from cache monitoring javascript code.
- Python-based adversarial AI tools to introduce artificial noise in cache to reduce classification rate of our attacker code.

Non-functional Requirements

- Must not increase system overhead by more than 25%.
- Attacker code must be able to identify with an accuracy rate of 80%.
- Defense code must lower accuracy of attacker code by at least 60%.

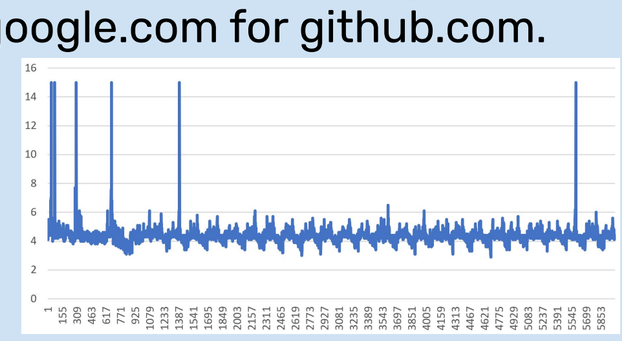
Adversarial AI Approach

- Explainable AI distance metrics for every input-output class combination created using modified versions of saliency map algorithm.
- Distance is measured by calculating the amount of gradients required to classify sample class as another

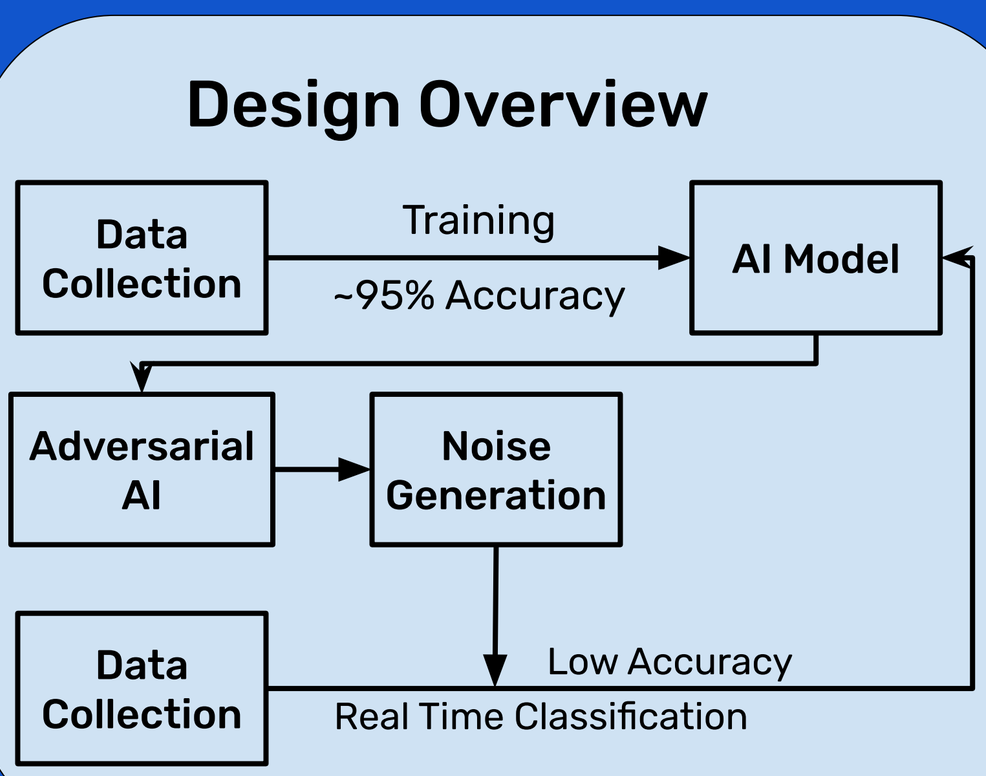


Javascript to Generate Noise

- Possible options were loading high resolution images, large text files, or websites with lots of data.
- Found that github.com requires lots of processing power to load.
- By using javascript to visit that website multiple times we were able to generate high amounts of noise at specific points.
- This leads to AI mistaking websites like google.com for github.com.

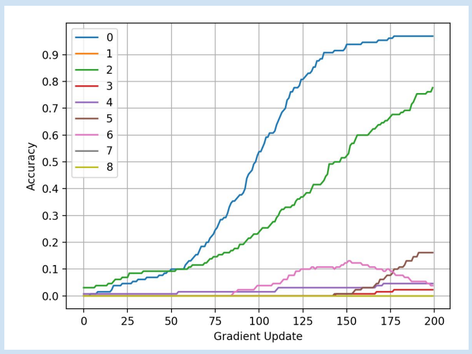


Design Overview



Testing

Gradients have been added for every input-output combination to measure distances of each class.



Previous Literature

❏ Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, Yuval Yarom (2019) Robust website fingerprinting through the cache occupancy channel. USENIX Security 2019