A Artifact Appendix

A.1 Abstract

We demonstrate how targeted deanonymization attacks performed via the CPU cache side channel can circumvent browser-based defenses. The attack framework we show is able to overcome the limitations of prior work, such as assumptions on the existence of cross-site leaks. As a result of this attack, the attacker is able to learn whether a specific individual visits the attacker-controlled website – a potentially serious privacy violation.

When a user visits the attacker-controlled website, the website uses an iframe, popunder, or tabunder to request a resource from a third-party website (i.e., the "leaky resource"). The response to this request, as well as the cache activity it generates in the user's system, differs depending on the user state on the third-party website. An attacker monitoring the CPU cache side channel can analyze the cache patterns and learn whether the leaky resource was loaded successfully in the browser or not, and use this information to learn the identity of the visiting user. The attack can be scaled to identify thousands of users.

The artifact repository is hosted at GitHub and evaluations are performed on Google Colab. The reviewers should run the provided scripts on Google Colab. To support the feasibility of the attacks and the defense proposed in the paper, the results should be similar to Figure 5 and Table 1, 2 and 6 of the paper.

A.2 Artifact check-list (meta-information)

- **Data set: dataset.zip**
- **Run-time environment: Google Colab**
- **How much disk space required (approximately)?: 200MB**
- **How much time is needed to complete experiments (approximately)?: one hour**
- **Archived (explicitly provide DOI or stable reference)?: https://github.com/leakuidatorplusteam/artifacts.git commit ID: 78bae165e0dbcdeb245b19a1f5b75a191de92fc3**

A.3 Description

We submit for **Artifacts Available**, **Artifacts Functional** and **Results Reproduced** badges.

A.3.1 How to access

```
git clone git@github.com:
leakuidatorplusteam/artifacts.git

cd artifacts
git checkout 78bae165e0dbcdeb245b19a1f5b75a191de92fc3
```

A.3.2 Hardware dependencies

To collect additional traces, one of the following systems are required:

```
Dell Latitude - Intel Core i7 7820HQ

Mac mini - Apple M1 8-Core

MacBook Pro - Intel Core i7 3540M
```

A.3.3 Software dependencies

To collect additional traces, one of the following systems are required:

```
Windows 10 Pro 20H2 - Chrome 96.0

macOS Big Sur 11.4 - Chrome 96.0

macOS Catalina 10.15.7 - Safari 15.0
```

A.3.4 Data sets

dataset.zip file is available at the root directory of the artifacts repository hosted at GitHub.

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

Git software can be used to get local access to the repository. Reviewers need to have Google accounts to access Google Colab and also to share resources privately.

A.5 Evaluation and expected results

Here we describe the step by step instructions of two phases of the evaluation. In the first phase, we demonstrate how the dataset we already collected can be used to train the classifiers and determine the accuracy of attacks: [1] [2]

1. Open https://colab.research.google.com/

2. From your local copy of artifacts repository, upload the USENIX_Artifact_Evaluation/cache_demo.ipynb file in Google Colab and open it
3. On the left side of Google Colab interface, click on “Files”, then “Upload to session storage”, and choose the *dataset.zip* file from your local copy of the artifacts repository
4. From the menu on top of Google Colab interface, click on “Runtime”, then “Run all”, and wait until it is finished

After finishing these 4 steps, the reported results are as follows:

- The code block with comment starting with "## [Single Target Attacks]" shows the prediction accuracy on the dataset using LR (logistic regression classifier), MSE (mean squared error), and FastDTW (fast dynamic time warping). These results correspond to the results reported on Table 1 and 6
- The code block with the comment starting with "## [Chrome Android]" shows the results for experiments with Android Chrome
- The code block with the comment starting with "## [Old Defense (Leakuidator)]" shows the results for experiments with old defense prior to the modifications suggested in this paper
- The code block with the comment starting with "### [Multi Target Attacks]" shows the results for experiments with multi target attacks, reported in Table 2 of the paper
- The code block with the comment starting with "## [Average and Attack Accuracy plots]" correspond to Figure 5 of the paper.

In the second phase, we provide a step by step instruction to demonstrate how the attack page collects the cache traces and uses them for prediction. To run the attack from scratch, reviewers can collect traces using one of three systems Win-Chrome, Mac-Intel-Safari, or Mac-M1-Chrome detailed in Table 4 of the paper, using the respective attack pages at USENIX_Artifact_Evaluation directory. To customize the targeted deanonymization attack demo for a target user of your choice, do the following:

1. Login to the attacker Youtube account (e.g., attacker@gmail.com) at youtube.com
2. Upload two private videos of at least 1 second duration in the attacker Youtube account
3. Write down the identifier of the private videos you created, called **[video_id_1]** and **[video_id_2]**
4. Share **[video_id_1]** privately only with the targeted victim you’d like to track (e.g. victim@gmail.com) and **[video_id_2]** privately only with another attacker account (e.g. attacker_second_id@gmail.com)
5. Prepare the state dependent URL as follows: "https://www.youtube.com/embed/**[video_id_1]** ?rel=0&autoplay=1&mute=1"
6. Prepare the URL for the non-target state as follows: "https://www.youtube.com/embed/**[video_id_2]** ?rel=0&autoplay=1&mute=1"
7. Prepare two attack pages page_1.html and page_2.html and change the "State-Dependent-URL" string in the source code of the attack pages to these two URLs: page_1 points to the URL at step 5 and page_2 points to the URL at step 6
8. Host the attack pages on a web server (either local or remote). In particular, do not run the attack pages as local files (i.e., not served by a web server)
9. Log out of the attacker’s youtube account, and login to the victim’s youtube account
10. Open two tabs in the browser. The first tab points to page_1 resembling the target state, and the second tab points to page_2 resembling the non-target state. Record the traces first in the target tab, then in the non-target tab, then again target tab, then non-target tab, ..., and repeat this at least 100 times. (to make the experiment easier, instead of manually performing this experiment, customize and use the scripts at the "automation_scripts" folder)
11. Put the collected traces into the template.json file (100 target traces and 100 non-target traces)
12. Open https://colab.research.google.com/
13. Upload the USENIX_Artifact_Evaluation/test.ipynb file to Google Colab and open it
14. On the left side of Google Colab interface, click on “Files”, then “Upload to session storage”, and choose the template.json file that contains your collected traces
15. Set the sweep and interval parameters as suggested in the comments
16. From the menu on top of Google Colab interface, click on “Runtime”, then “Run all”, and wait until it is finished

After finishing these steps, an average plot is generated. It should be somewhat similar to Figure 5 in the paper, demonstrating the differences between the two states. Also, accuracy of the logistic regression classifier is reported.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220119.