# **Drawn**Apart

A Deep-Learning Enhanced GPU Fingerprinting Technique

**Tomer Laor**[1], **Naif Mehanna**[2], Antonin Durey[2], Vitaly Dyadyuk[1], Pierre Laperdrix[2], Clémentine Maurice[2], Romain Rouvoy[2], Walter Rudametkin[2], Yossi Oren[1], Yuval Yarom[3]

(1) Ben-Gurion University of the Negev
(2) Univ. Lille / Inria / CNRS
(3) University of Adelaide

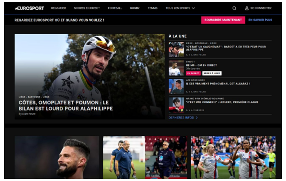
# A possible use case

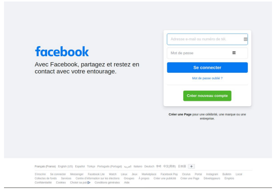
*Unethical advertiser*

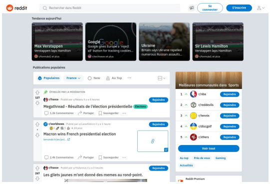
Jack likes sports

Jack uses facebook & reddit

*Let’s show sport ads in there !*





# A possible use case

*Unethical advertiser*

Jack likes sports

Jack uses facebook & reddit

*Let's show sport ads in there !*


*Jack disabled his cookies*

*He made sure to randomize his browser fingerprint*

*He did not log in sensitive website*

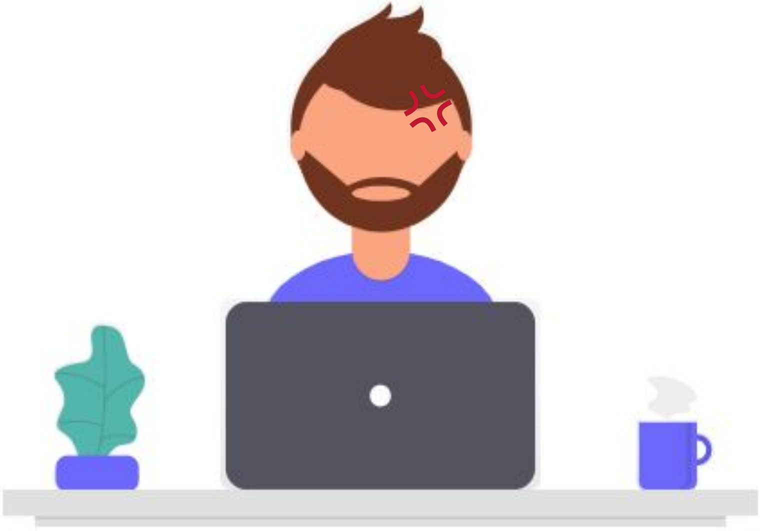
# A possible use case

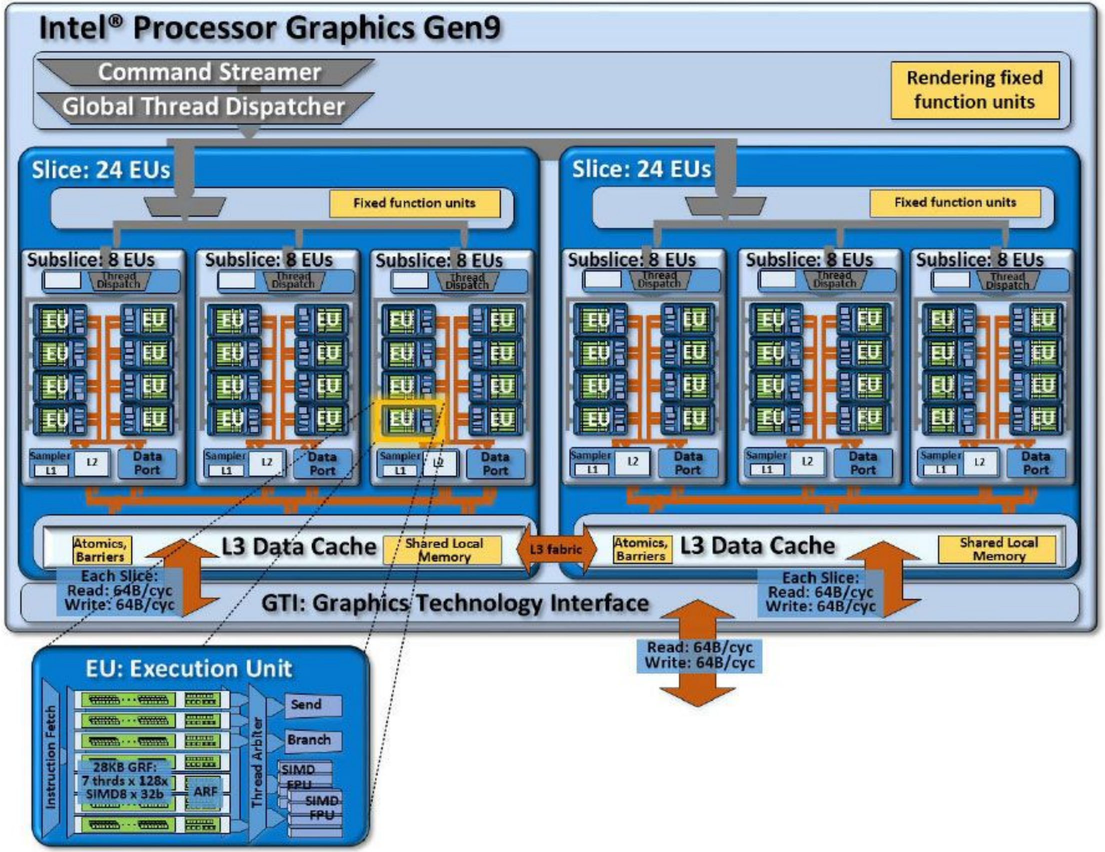

How did the advertiser manage to track Jack ?

# A possible use case

*Unethical advertiser*

***We used innovative hardware fingerprinting!***






# What can we extract from it?

# Hypothesis

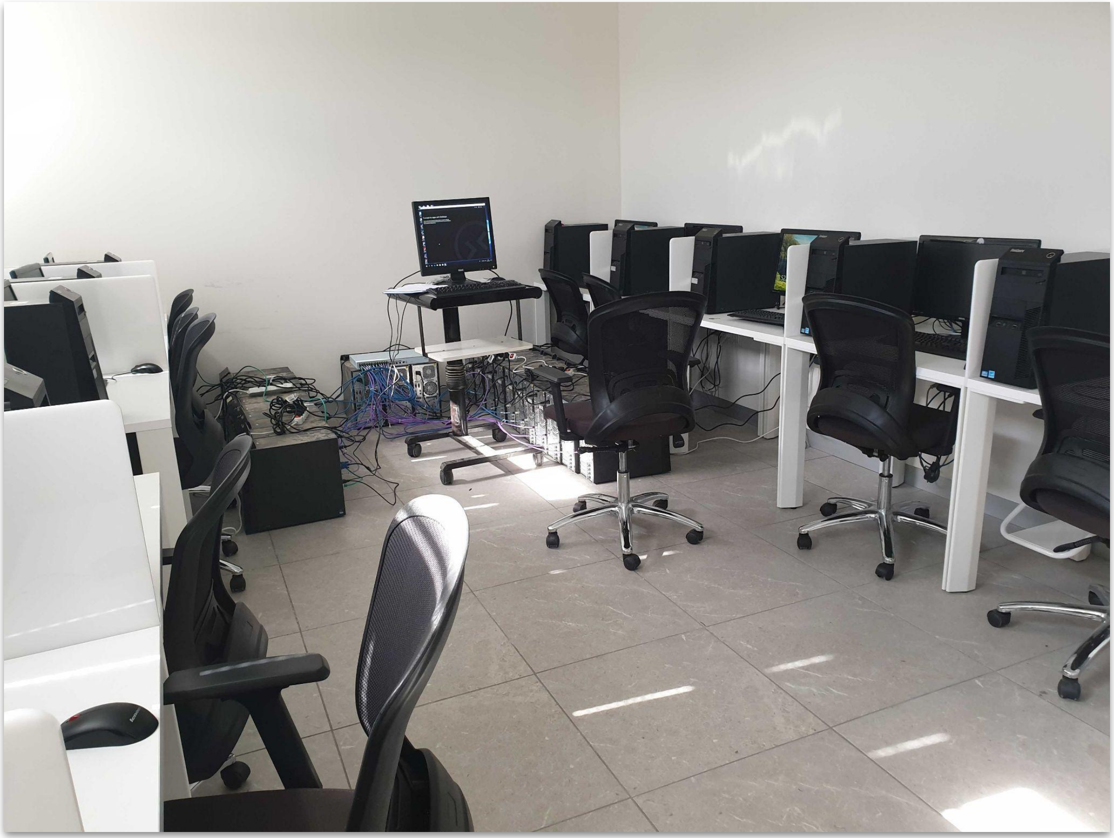
Each GPU, even from the same model, shows **differences** on some scale.

# Verifying the Hypothesis

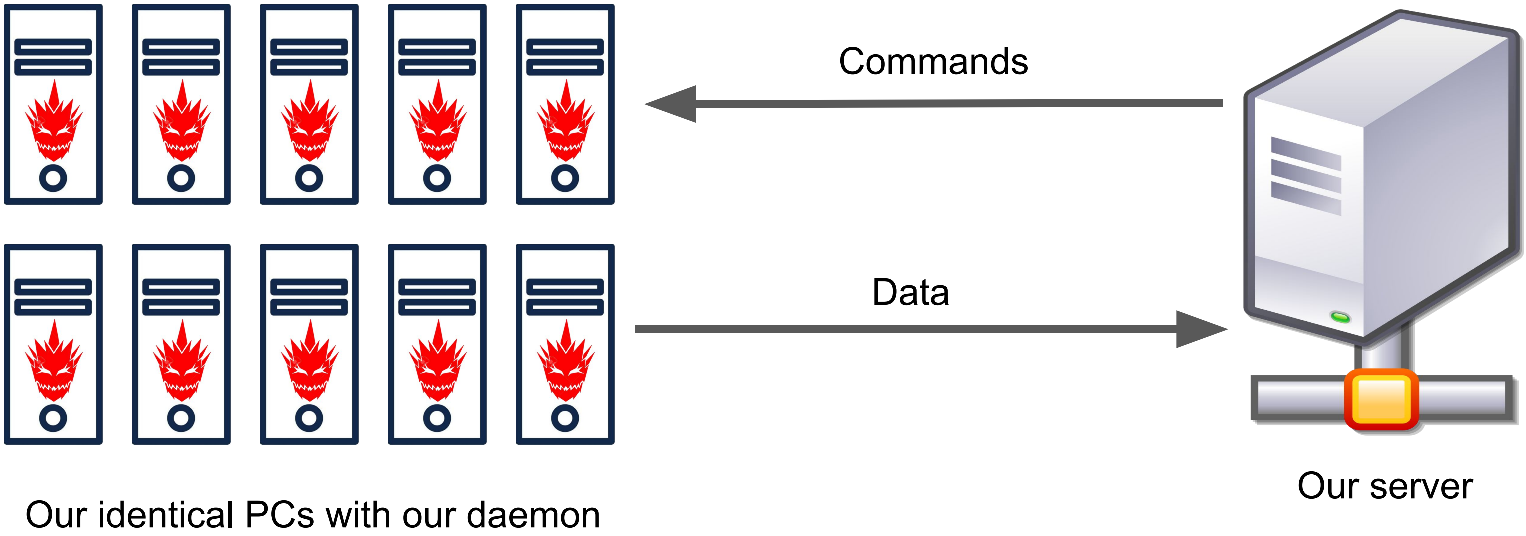
We need to be able to:

- Experiment with different scripts interacting with the GPUs.
- Run the same code on multiple machines with the same software and hardware.
- Have the multiple machines in the same environment (temperature, pressure, etc…)

# The Setup



# The Setup



# Web Environment

We need to use WebGL to run code on the GPU from a web page.

**WebGL doesn’t have mutexes.**

# Web Environment - Offscreen

```
def drawn_apart_offscreen():
  times = []
  for vertices_to_stall in power_set(vertices_num):
    start_time = time.now()
    apply_in_parallel(vrtx_to_render=>render_vertex(vertices_to_stall, vertex_to_render))
    end_time = time.now()
    times.append(end_time - start_time)
  return times


def render_vertex(vertices_to_stall, vertex_to_render):
    if vertex_to_render not in vertices_to_stall:
        render(color='green')
    else:
        render(color=intensive_compute())
```

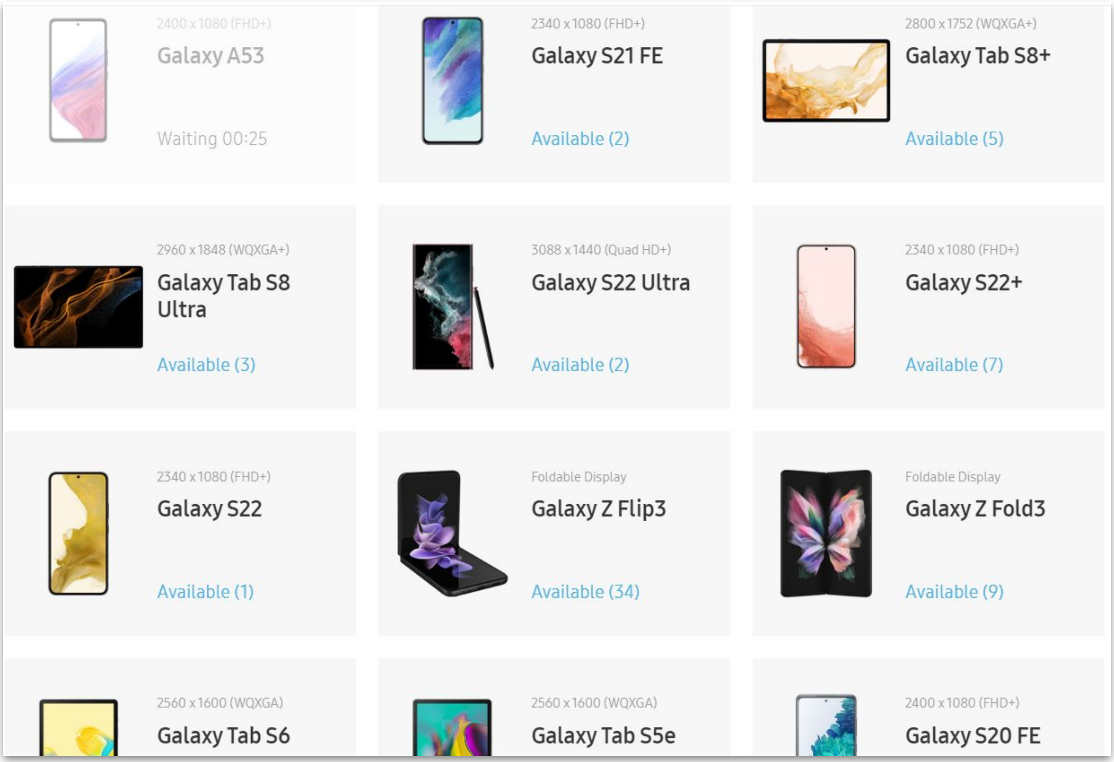
# Web Environment - Offscreen

These traces were classified using Random Forest.

# Web Environment - Offscreen Results

| Device Type | GPU | Device Count | Base Rate (%) | Accuracy (%) | Gain |
|---|---|---|---|---|---|
| Intel i5-3470 (GEN 3 Ivy Bridge) | Intel HD Graphics 2500 | 10 | 10 | 36.3±1.6 | 3.6 |
| Intel i5-4590 (GEN 4 Haswell) | Intel HD Graphics 4600 | 23 | 4.3 | 63.7±0.6 | 14.7 |
| Intel i5-8500 (GEN 8 Coffee Lake) | Intel UHD Graphics 630 | 15 | 6.7 | 55.5±0.8 | 8.3 |
| Intel i5-10500 (GEN 10 Comet Lake) | Nvidia GTX1650 | 10 | 10.0 | 70.0±0.5 | 7.0 |
| Apple Mac mini M1 | Apple M1 | 4 | 25.0 | 46.9±0.4 | 1.9 |

# Can DrawnApart Work On Mobile Phones?




SAMSUNG
Remote Test Lab

# Web Environment - Results

We swapped the hard drives of 2 devices in the Intel i5-3470 set.

**Spoiler**: We were still able to identify the correct device using DrawnApart!

What happened ?

Scan to watch the cyb3r video!

# Web Environment - Hypothesis

**WebGL deterministically assigns execution units to vertices.**

# AmIUnique Integration

AmIUnique has a Chrome **extension that follows changes** in your **browser fingerprint** over time.

**We integrated the DrawnApart offscreen method with AmIUnique in order to gather data in a real world setting**

# AmIUnique Integration

- Make sure that users won’t feel slowdowns
- Select the best stall function for the in-the-wild settings
- Ensure that it will support all the different configurations that can occur in the wild.

# Large Scale Experiment - Dataset

The dataset contains **~370,000 fingerprints** collected from **~2,500 unique devices** through the AmIUnique platform.

Each collection includes 7 traces.

AM I UNIQUE ?

https://amiunique.org

# Large Scale Experiment - ML

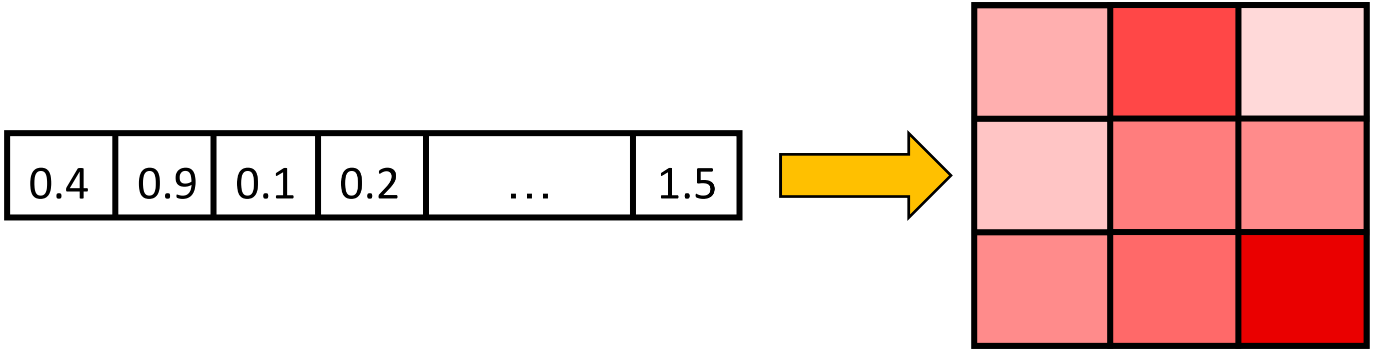
We first tried to use Random Forest.

But… training on ~2,500 labels required an extensive amount of RAM → **Not ideal in a real world setting**

**We tried to make clusters of devices using the canvas hash and renderer string, but the story wasn't compelling enough…**

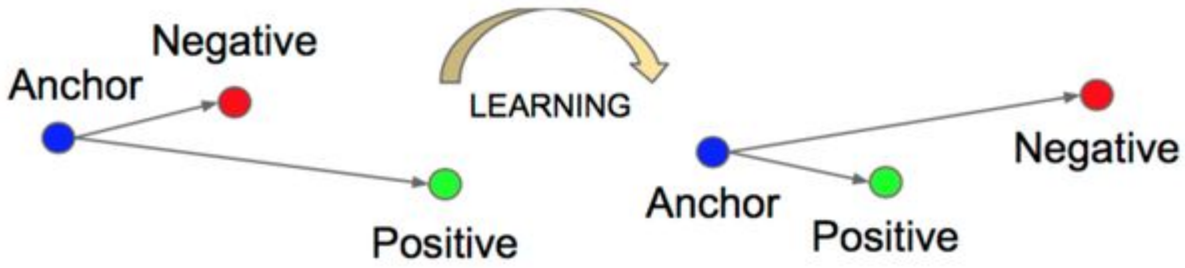
# Large Scale Experiment - ML

Neural networks have the expressive power that we need, with a reasonable runtime and RAM usage.

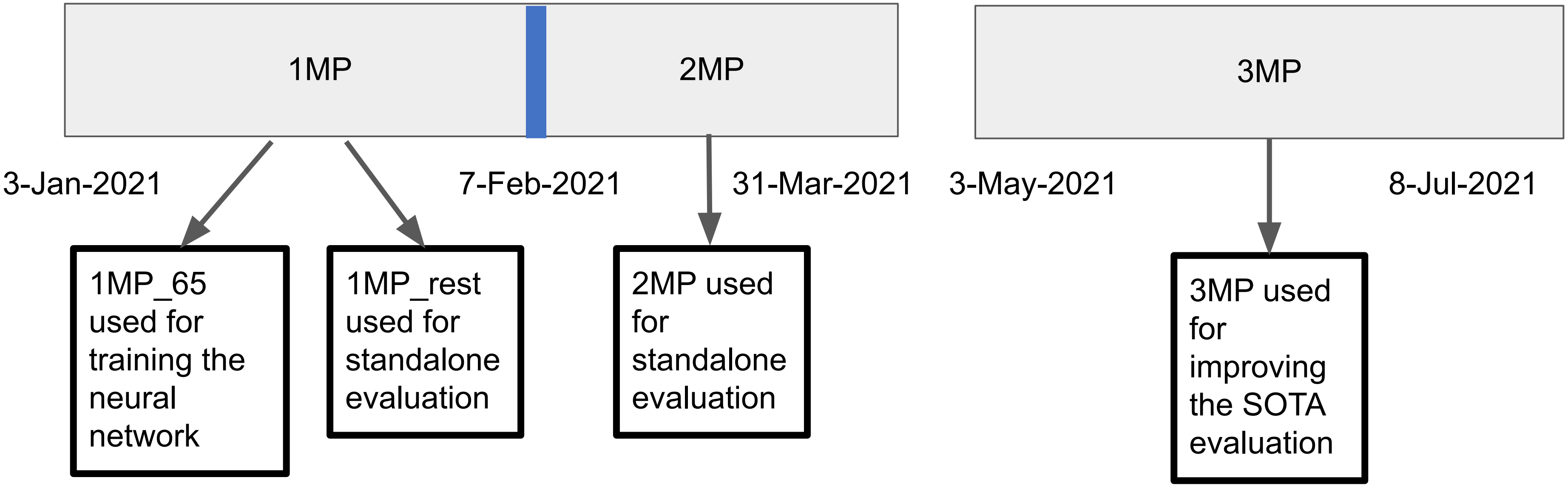


# Large Scale Experiment - ML

We trained a CNN with **semi-hard triplet loss** to **map the original feature space into a lower dimension** Euclidean space.



# In The Wild Dataset Split



# Large Scale Experiment - Grid5000

Grid5000 is a **big cluster** that let us access machines with powerful GPUs.

**We trained the DrawnApart deep learning solution on Grid5000.**

https://grid5000.fr

# Tested neural network architectures

Prior to picking the *Convolutional Neural Network*, we tried various methods:

**Vision Transformers** → lower accuracy overall - harder to optimize

**Training using a Siamese Networks setting** → harder to optimize

**LSTM networks** → Significantly lower accuracy

**DeepAR** → We considered a trace to be a time-serie - bad accuracy

# What Would an Attacker Do?

1) Gather a lot of data from a lot of users

2) Train an embedding CNN

3) **In production**: transform each incoming trace using the CNN and **compute the distance** to the existing embeddings.

# Evaluation: In-the-Wild Conditions

- We have a lot of data at hands
- Results are good thanks to the amount of data

| Top-10 Accuracy | Top-5 Accuracy | Top-1 Accuracy | Rate Top-1 Base | Nu. of Traces For Memorizing |
|---|---|---|---|---|
| 67.15% | 55.09% | 28.28% | 1.0% | 420K~ |

# What Would an Attacker Do?

1) Same attacker - different website with different users.

2) The attacker already have a trained neural network

3) In production: **1-shot learning**

**This is a harder task!**

# Evaluation: K-Shot Conditions

- A small amount of data

- Results are impressive even though the task is harder.

| Top-10 Accuracy | Top-5 Accuracy | Top-1 Accuracy | Top-1 Base Rate | Nu. of Traces For Memorizing | Method |
|---|---|---|---|---|---|
| 19.95% | 14.10% | 5.44% | 0.00%~ | 14K~ | Shot-1 |
| 26.75% | 19.34% | 7.11% | 0.00%~ | 59K~ | Shot-5 |
| 31.09% | 22.77% | 9.22% | 0.00%~ | 100K~ | Shot-10 |

# Improving the State-of-the-Art

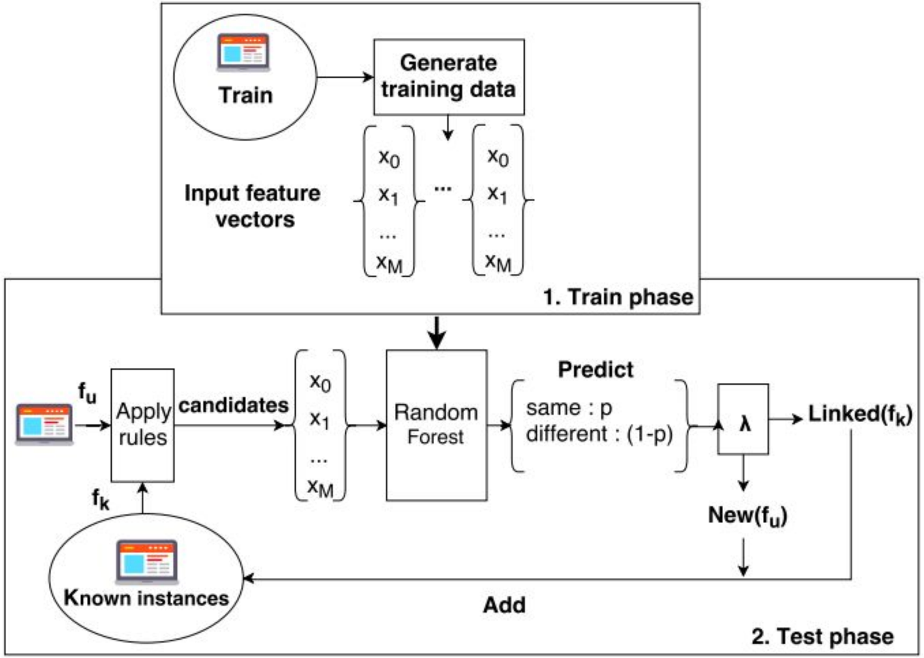
- FP-Stalker: Published at **S&P 2018**

- At the time, it was shown that browser fingerprinting was efficient for identification but **not for long-term tracking**

- **FP-Stalker** showed that browser fingerprinting could be used for long term tracking

Vastel, A., Laperdrix, P., Rudametkin, W., & Rouvoy, R. (2018, May). Fp-stalker: Tracking browser fingerprint evolutions. In *2018 IEEE Symposium on Security and Privacy (SP)* (pp. 728-741). IEEE.

# Improving the State-of-the-Art




(b) Hybrid variant of FP-STALKER. The training phase is used to learn the probability that two fingerprints belong to the same browser instance, and the testing phase uses the random forest-based algorithm to link fingerprints.

FP-Stalker has two main steps:

- Rule-based filtering
- Machine learning inference

# FP-Stalker & DrawnApart

| | |
|---|---|
| Collection timestamp | User Agent (HTTP) |
| Hashed Canvas (JS) | Do Not Track (JS) |
| Language (HTTP) | Cookies (JS) |
| Plugins (JS) | Local (JS) |
| Renderer (JS) | Flash-based attributes |
| Screen Resolution (JS) | |
| Timezone (JS) | |

+

Processed DrawnApart trace

# Adapting FP-Stalker

- Between 2017 and today, the web ecosystem **changed !**
- Flash became **unsupported** by all major browsers since 2021
- FP-Stalker **relied on flash-based attributes** for its rule-based step →it **couldn't be applied** in the current web

**We adapted FP-Stalker to the current web, while ensuring that the results remained on par with the paper.**

# Adapting FP-Stalker

1) Understanding **the logic behind FP-Stalker** & inspecting the existing code

2) Identifying **what can be optimized** →We identified **several bugs that could impact the accuracy** and optimized the code logic and readability

3) **Comparing our adapted version of FP-Stalker to its original algorithm** on the provided dataset

# Adapting FP-Stalker

In order to introduce DrawnApart into FP-Stalker, we had to **identify the ideal position** in the algorithm.

We noticed that **the Machine learning step classified only a few percentage** of the traces due to the generated threshold being **too high**.

**Algorithm 2** Hybrid matching algorithm

```
function FINGERPRINTMATCHING(F, f_u, λ)
    rules = {rule_1, rule_2, rule_3}
    exact ← ∅
    F_ksub ← ∅
    for f_k ∈ F do
        if VERIFYRULES(f_k, f_u, rules) then
            if nbDiff = 0 then
                exact ← exact ∪ ⟨f_k⟩
            else
                F_ksub ← F_ksub ∪ ⟨f_k⟩
            end if
        end if
    end for
    if |exact| > 0 then
        if SAMEIDS(exact) then
            return exact[0].id
        else
            return GENERATENEWID()
        end if
    end if
    candidates ← ∅
    for f_k ∈ F_ksub do
        ⟨x_1, x_2, ..., x_M⟩ = FEATUREVECTOR(f_u, f_k)
        p ← P(f_u.id = f_k.id | ⟨x_1, x_2, ..., x_M⟩)
        if p ≥ λ then
            candidates ← candidates ∪ ⟨f_k, p⟩
        end if
    end for
    if |candidates| > 0 then
        c_h1, p_h1 ← GETCANDIDATESRANK(candidates, 1)
        c_h2, p_h2 ← GETCANDIDATESRANK(candidates, 2)
        if SAMEIDS(c_h1) and p_h1 > p_h2 + diff then
            return candidates[0].id
        end if
        if SAMEIDS(c_h1 ∪ c_h2) then
            return candidates[0].id
        end if
    end if
    return GENERATENEWID()
end function
```

# Adapting FP-Stalker

FP-Stalker uses the **average and maximum tracking time** to quantify the performances of its algorithm.

**Average tracking time**: For a given device, how long can we track it on average?

We used both metrics to quantify the improvement of DrawnApart on FP-Stalker

# Adapting FP-Stalker

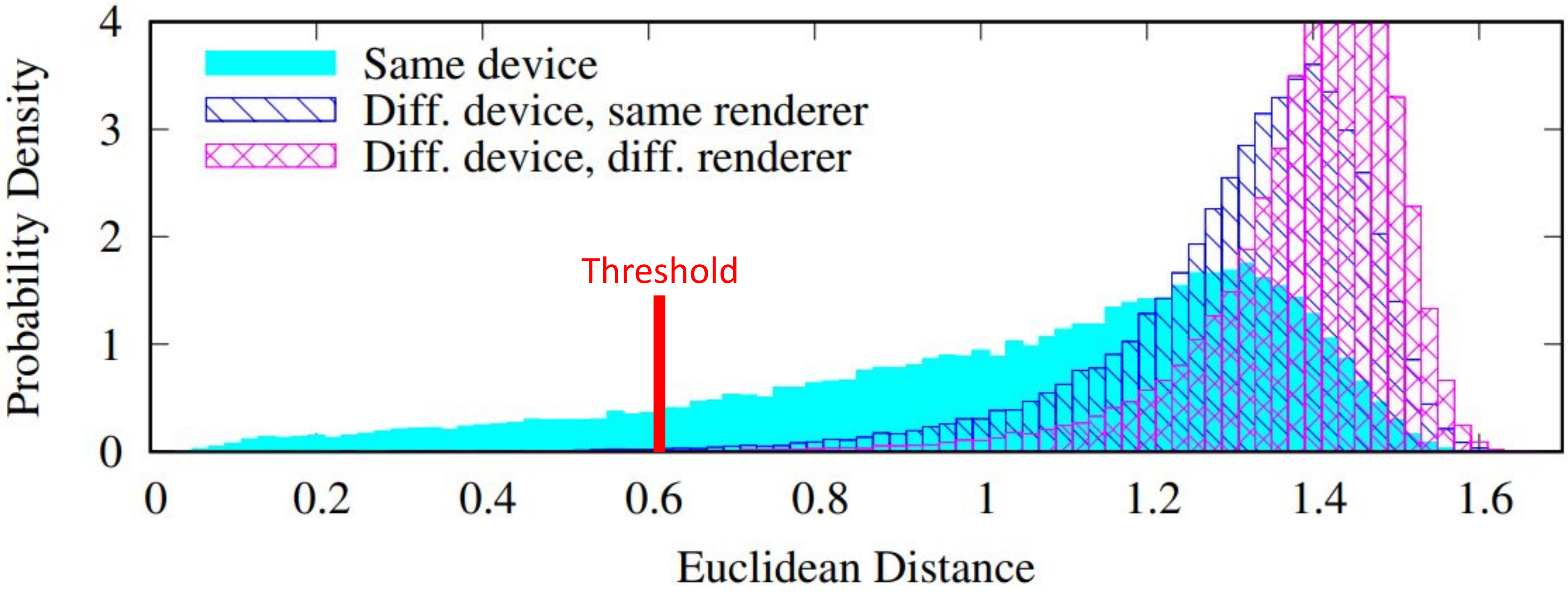
We integrated our deep-learning pipeline by **short-circuiting the machine learning step**.

If the Cosine distance between two traces is below the threshold, we conclude the search.

**Algorithm 1:** Hybrid matching algorithm with the DRAWNAPART addition highlighted in red

**Function** *FingerprintMatching* ($F$, $f_u$, $\lambda$, $\epsilon$)
  **for** $f_k \in F$ **do**
    **if** *FingerPrintHasDifferences*($f_k$, $f_u$, *rules*) **then** $F_{ksub} \leftarrow$ exact $\cup < f_k >$ ;
    **else**
      exact $\leftarrow$ exact $\cup$ ¡$f_k$¿
    **end**
  **end**
  **if** $|exact| > 0$ **then**
    **if** *SameIds(exact)* **then** **return** *exact[0].id* ;
    **else** **return** *GenerateNewId()* ;
  **end**
  **for** $f_k \in F_{ksub}$ **do**
    cosine_sim $\leftarrow$ GetSimilarity($f_u$.*avg_embedding*, $f_k$.*avg_embedding*);
    **if** *cosine_sim* $> \epsilon$ **then**
      **return** $f_k$.*id*
    **end**
    $< x_1, x_2, .., x_m >$ = FeatureVector($f_u$, $f_k$);
    p $\leftarrow$ P($f_u$.*id* = $f_k$.*id* $| < x_1, x_2, .., x_m >$)
    **if** $p \geq \lambda$ **then**
      candidates $\leftarrow$ candidates $\cup < f_k, p >$
    **end**
  **end**
  **if** $|candidates| > 0$ **then**
    **if** $|GetRankAndFilter(candidates)| > 0$ **then** **return** *candidates[0].id* ;
  **end**
  **return** *GenerateNewId()*

# Analyzing DrawnApart



# Adapting FP-Stalker

FP-Stalker was criticized for its algorithm being **too slow**.

We noticed that our updated version with DrawnApart mitigated this limitation by **significantly reducing the execution time**.

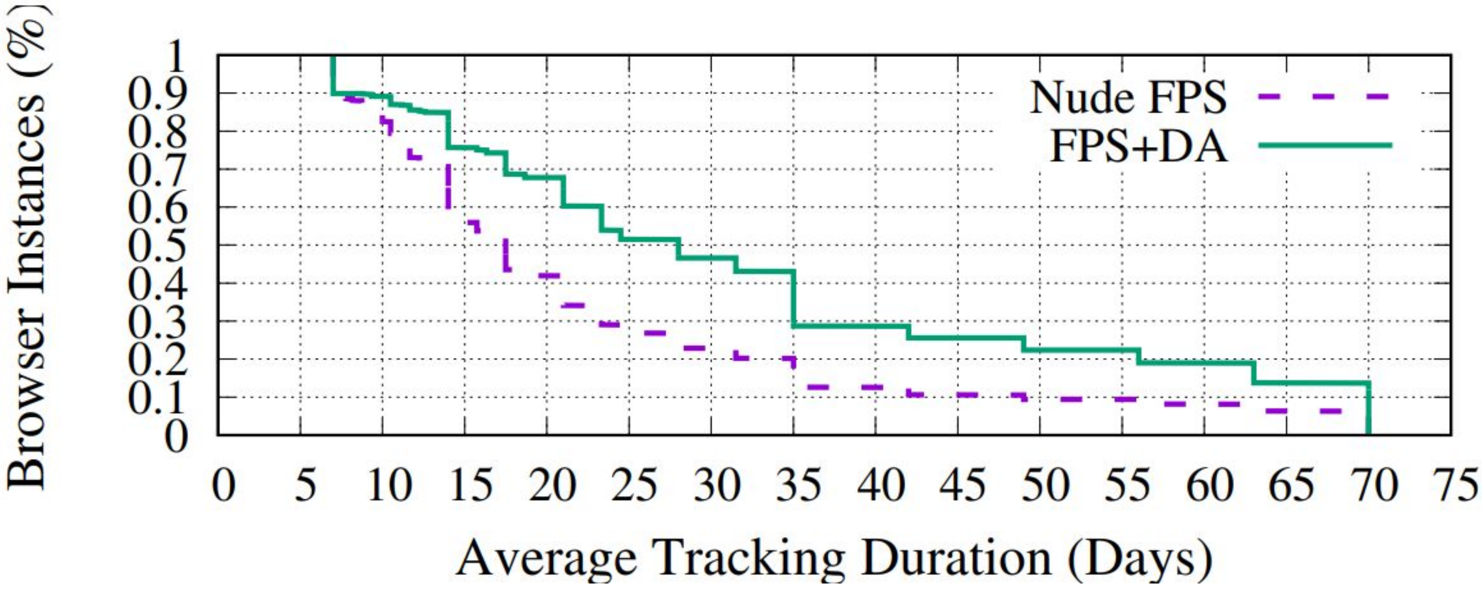
**Algorithm 1:** Hybrid matching algorithm with the DRAWNAPART addition highlighted in red

```
Function FingerprintMatching (F, f_u, λ, ε)
  for f_k ∈ F do
    if FingerPrintHasDifferences(f_k, f_u, rules)
      then F_ksub ← exact ∪ < f_k > ;
    else
      exact ← exact ∪ ¡f_k¿
    end
  end
  if |exact| > 0 then
    if SameIds(exact) then return exact[0].id ;
    else return GenerateNewId() ;
  end
  for f_k ∈ F_ksub do
    cosine_sim ←
      GetSimilarity(f_u.avg_embedding,
      f_k.avg_embedding);
    if cosine_sim > ε then
      return f_k.id
    end
    < x_1, x_2, .., x_m > = FeatureVector(f_u, f_k);
    p ← P(f_u.id = f_k.id | < x_1, x_2, .., x_m >)
    if p ≥ λ then
      candidates ← candidates ∪ < f_k, p >
    end
  end
  if |candidates| > 0 then
    if |GetRankAndFilter(candidates)| > 0 then
      return candidates[0].id ;
  end
  return GenerateNewId()
```

Li, S., & Cao, Y. (2020, October). Who touched my browser fingerprint? A large-scale measurement study and classification of fingerprint dynamics. In *Proceedings of the ACM Internet Measurement Conference* (pp. 370-385).

# Improving FP-Stalker

Differences in Average Tracking Time between FP-Stalker (*Nude FPS*) and FP-Stalker with DrawnApart (*FPS+DA*)



# Thank You
# Let’s discuss !

Naif Mehanna
naif.mehanna@univ-lille.fr

GitHub repository

Tomer Laor
tomerlao@post.bgu.ac.il