Postdoc proposal

Advanced Script analysis to prevent Online Tracking

1 Contact

- Pierre Laperdrix (Full-time researcher, CNRS, pierre.laperdrix@univ-lille.fr)

2 Research team

The postdoc will join the Spirals (`https://team.inria.fr/spirals`) project-team between the University of Lille and Inria, within the UMR CRIStAL Laboratory.

Université de Lille
UMR CRIStAL
Bâtiment M3, Université de Lille 1,
59655 Villeneuve d'Ascq – FRANCE

Inria Lille - Nord Europe
Parc Scientifique de la Haute Borne
40, avenue Halley - Bat. B, Park Plaza
59650 Villeneuve d'Ascq – FRANCE

3 Postdoc topics

Positioned in the context of web security and privacy, this postdoc will focus on any of the following topics.

3.1 Detecting tracking entities

When users go online, they see what is being rendered by their browser but they do not suspect the amount of scripts being executed in the background. Some scripts are benign as they connect a button from the UI to its corresponding action but others are more nefarious and want to track users' every move. There is clear lack of control of the scripts sent to us by online servers as we have no true understanding of what each of them does.

Finding the purpose of online scripts One possible topic of this postdoc is to map automatically each script to its respective functionality so that we can design advanced defense mechanisms that go much further than the simple ad blocking based on name matching we see today. Knowing if a script is essential to a page or if it is only useful for tracking can make or break a modern tracking protection. To achieve this goal, it is possible to rely on the following techniques:

- **Information flow:** In the studies focused on web tracking, the authors look at either what is being accessed by the browser or what is being sent by it but not at the link between the two. When information is collected by calling a browser API, it can be changed, mixed, hashed or even encoded by some other functions. Looking only at both end of the spectrum inhibits our ability to fully understand what a script is doing and if it can be nefarious or totally benign. One way of performing information flow in a web browser is to have a *taint engine*. It assigns labels to specific objects in memory and propagates if it is modified or accessed by others. This way, if the screen resolution is accessed, stored in a object, hashed and then sent over the network for fingerprinting purposes, having this propagation of taint will help follow the life of this information until it is sent to its final destination. A major challenge to have a functioning taint engine is to integrate such concept into a modern browser code base and maintain it over time. The sheer size of a modern web browser in terms of code poses both scientific and engineering hurdles that this project will aim to answer.

- **Machine learning:** To identify scripts with similar functionality, it is essential to use machine learning as it can go beyond what one can do with hard coded heuristics. The major scientific challenge here is to find the right dimensions to obtain a meaningful clustering of scripts. Some of these dimensions may include using the script's structure, its flow of information or the information sent over the network. Having a working taint engine has described above will help in obtaining more accurate results.

- **Deobfuscation:** A lot of scripts on the Internet are obfuscated and minified. While it may help reducing the overall bandwidth used by a website as less data has to be transmitted, it hampers our understanding of what these scripts do as they become simply non-human readable. Being able to deobfuscate scripts on the web with the help of both information flow and machine learning can help determine if a script has been optimized or if it is hiding some malicious code.

Detecting browser fingerprinting Being able to map a script to its respective functionality will push the detection of browser fingerprinting forward on the web. Today, we see a lot of scripts accessing device-specific properties but we have no understanding if this information is used to optimize the user experience or if it is sent over the network to build a profile of the user. Having more data of what a script is doing will help make a more informed decision to block it or not.

One important aspect of fingerprinting are the attributes used to build a fingerprint of the user's device. As browsers are evolving at a frantic pace, new APIs are introduced every few years to provide new features to users. However, despite careful attention from standard organization like W3C, some attributes introduce new differences between users that could

be exploited to build even stronger browser fingerprints. In order to make sure that new APIs are free of such differences, it is possible to realize the two following tasks:

- **Building a testing workbench:** Having a testing workbench can be very valuable to constantly test browsers for detectable differences between them. A workbench will enable the discovery of fingerprintable attributes introduced by new APIs but can also help in the discovery of attributes that are already present in our browsers but that are still dormant.

- **Backwards flow analysis:** Using a precise information flow, it is possible to go backwards in the analysis of information flow to identify APIs that were never labelled as containing device information. For example, if a script puts in the same object fingerprinting information along with data from a newly introduced API, it is important to analyze what this new data contains and if it can abused to enrich the fingerprint of a device.

In the end, having a better understanding of what a script does will provide better decision making when it comes to protecting web users without impacting the user experience.

3.2 Controlling the information sent to external servers

The second set of topics for this postdoc is to protect users by restricting the information sent to external servers. This can be done by either blocking nefarious scripts or by restricting the information accessible through the different browser APIs.

Blocking scripts and identifying page breakage Essential functionality of a website can be sometimes intertwined with a tracking mechanism so blocking it can transitively "break" a webpage. We define "page breakage" as an undesirable behavior on a webpage. This includes, but is not limited to, page slowdowns, page freezes, page crashes and errors, page display issues, among others.

Here, we want to automatically identify page breakage, find its source and provide a potential fix to it. To achieve this objective, it is possible to explore the following areas:

- **Collecting data at different levels:** To identify problems and pinpoint their sources, we need to combine different types of data to have a perfect understanding of the execution environment where the breakage occurs. At the JS interpreter level, logs can help to identify errors and coding issues. At the rendering engine level, DOMs can highlight discrepancies in the structure of loaded HTML documents. At the network level, traces can show the presence of cookies and the loading of third-party components. At the UI level, screenshots of pages can be taken and user interaction can be recorded to identify loading errors or missing components.

- **Detecting breakage automatically:** By using the workbench described in the previous part, it is possible to test the same webpage with a wide variety of software and hardware configuration, with and without tracking protection, with and without specific browser extensions or with and without specific privacy settings. The discovery

can then be automated: is a page broken on all configurations or does it affect only a subset of machines? Does the DOM structure or the network activity vary significantly between two devices? In the end, having the right information at the right level is key to better understand potential page breakage and improve protection on the user's side.

Restricting device information accessible through browser APIs to limit browser fingerprinting Browser fingerprinting exists today because it relies on mechanisms that have existed since the beginning of the Internet to improve the user experience. At that time, computing resources and bandwidth were limited and imposed constraints of what a web page contained so that it simply worked on the user's device. Here, we want to investigate how much we can restrict and remove information contained in these legacy APIs to limit the use of fingerprinting online. Relying on the workbench described above, we can perform crawls of the web by blocking APIs used for fingerprinting and by removing device-specific information to identify potential page breakage. This will provide us with key information on how the web still relies on these pieces of data and it would inform us on how we could move forward toward a fingerprint-less privacy-focused Internet.