Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting

Soroush Karami, Panagiotis Ilia, Konstantinos Solomos, Jason Polakis
University of Illinois at Chicago, USA
{skaram5, pilia, ksolom6, polakis}@uic.edu

***Abstract*—With users becoming increasingly privacy-aware and browser vendors incorporating anti-tracking mechanisms, browser fingerprinting has garnered significant attention. Accordingly, prior work has proposed techniques for identifying browser extensions and using them as part of a device's fingerprint. While previous studies have demonstrated how extensions can be detected through their web accessible resources, there exists a significant gap regarding techniques that indirectly detect extensions through behavioral artifacts. In fact, no prior study has demonstrated that this can be done in an automated fashion. In this paper, we bridge this gap by presenting the first fully automated creation and detection of behavior-based extension fingerprints. We also introduce two novel fingerprinting techniques that monitor extensions' communication patterns, namely outgoing HTTP requests and intra-browser message exchanges. These techniques comprise the core of Carnus, a modular system for the static and dynamic analysis of extensions, which we use to create the largest set of extension fingerprints to date. We leverage our dataset of 29,428 detectable extensions to conduct a comprehensive investigation of extension fingerprinting in realistic settings and demonstrate the practicality of our attack. Our in-depth analysis confirms the robustness of our techniques, as 83.6% - 87.92% of our behavior-based fingerprints remain effective against a state-of-the-art countermeasure.**

Subsequently, we aim to explore the true extent of the privacy threat that extension fingerprinting poses to users, and present a novel study on the feasibility of inference attacks that reveal private and sensitive user information based on the functionality and nature of their extensions. We first collect over 1.44 million public user reviews of our detectable extensions, which provide a unique macroscopic view of the browser extension ecosystem and enable a more precise evaluation of the discriminatory power of extensions as well as a new deanonymization vector. We also automatically categorize extensions based on the developers' descriptions and identify those that can lead to the inference of personal data (religion, medical issues, etc.). Overall, our research sheds light on previously unexplored dimensions of the privacy threats of extension fingerprinting and highlights the need for more effective countermeasures that can prevent our attacks.

I. Introduction

As Internet connectivity continues to proliferate globally, reaching almost ubiquitous presence in many countries, a large fraction of our everyday activities have migrated to the web. While mobile apps generate a significant amount of traffic, browsers still mediate a large portion of our online activities. As a result, the evolution of websites from static resources to functionality-rich applications has also necessitated the evolution of browsers into complex platforms with a rich set of APIs and features. To improve user experience, browsers allow users to further personalize them and extend their functionality by installing extensions.

Apart from the obvious benefits for users [26], [38], [48], extensions also introduce a privacy risk. Due to the potential risk, browsers do not provide any mechanism that would allow a visited webpage to directly obtain the list of installed browser extensions. In practice, however, webpages can indirectly infer which extensions are installed [24], [44], [46], [47]. Once the list of installed extensions is obtained, it can be used as part of a user's device fingerprint and coupled with other browser [18], [33], [40] or system level [13] information, which can lead to the tracking of users across the web [8], [19]. Extensions may also directly leak sensitive information like visited pages and form data to third parties [51]. While Firefox and Safari have tried to prevent certain extension enumeration techniques, Chrome –the most popular browser with a market share of ~64% [53]– remains vulnerable.

In this paper, our motivation is twofold: to conduct a comprehensive exploration of automated extension enumeration techniques under realistic settings, and to understand the true extent of the privacy threat that extension fingerprinting poses to users (apart from facilitating browser fingerprinting and web tracking). To that end we build Carnus, a modular system that analyzes Chrome extensions statically and dynamically for creating fingerprinting signatures and inferring sensitive data. Our system employs four different techniques for detecting extensions; first, it leverages the straightforward technique of identifying extensions that expose unique web-accessible resources (WARs), which has been demonstrated in prior studies [24], [47] at a smaller scale. Next, we focus on the identification of extensions through the detection of unique behavior-based modifications to a page's DOM. While this approach has been proposed as a potential fingerprinting technique [52], *no* prior work exists on the automatic generation of behavioral fingerprints. Here we tackle this challenging task, detail our technical approach, and demonstrate our system's effectiveness at automatically creating and detecting fingerprints at scale.

We also introduce two new techniques for inferring the presence of extensions based on intra-browser and external communication. Specifically, we find that certain extensions broadcast messages for communicating with components within the page (e.g., injected JavaScript), which we use to build fingerprints. Similarly, extensions can also send HTTP

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.24383
www.ndss-symposium.org

requests to external servers to fetch resources. We conduct a crawl of the Chrome Web Store and are able to fingerprint 29,428 extensions using all these techniques, resulting in the largest and most complete fingerprinting study to date. To demonstrate the robustness of our techniques we evaluate the impact of a recently presented state-of-the-art countermeasure [55] and find that our system can still fingerprint 83.6% - 87.92% of the behavior-based extensions.

Subsequently we measure the tracking capability enabled by extension fingerprints. While prior work has conducted user studies on a smaller scale and using smaller sets of fingerprintable extensions [24], [52], our goal is to accurately gauge the usefulness of extension fingerprints under more realistic settings in terms of scale. Given the significant challenge of conducting very large user studies with actual participants, we identify an alternative data source that offers a unique view into the set of extensions that users install, thus enabling such an analysis. Specifically, we collect over 1.44 million publicly available reviews for the extensions that are fingerprintable by Carnus. Using this dataset we conduct an analysis of the unicity of installed browser extensions for over 1.16 million users, and explore the feasibility of a novel deanonymization attack. Our results show that extensions installed by users can be highly identifying; for instance, if an attacker detects 4 random extensions that are fingerprintable by our system, there is a 94.47% chance that she can uniquely identify the user and learn their name and profile picture. While this deanonymization attack is not applicable to all users, since not everyone writes reviews, it reveals a significant privacy risk that stems from a seemingly innocuous action, i.e., writing a review about a browser extension.

Finally, we investigate the feasibility of attacks that infer user information based on the intended functionality of the discovered extensions. While not all extensions reveal sensitive information about the user (e.g., an ad-blocker), other extensions can explicitly or implicitly disclose information that is personal (e.g., ethnicity) or sensitive (e.g., religion). Our analysis shows that at least 18,286 of the extensions reveal such information. When considering the most sensitive types of information, we find that 147, 116, and 387 extensions expose the user's medical/health conditions, religion and political inclinations, respectively. Also, we find that the extensions that expose such sensitive information have been downloaded almost 2.5 million times. These findings highlight the privacy risk of users installing browser extensions, as websites and third-party services can surreptitiously infer personal and sensitive information. In summary, our research contributions are:

- We develop Carnus, a system that combines dynamic and static techniques for automatically analyzing extensions, and demonstrate the *first* automated creation and detection of behavior-based fingerprints. We provide a detailed technical description of our novel framework, which fully automates the entire fingerprinting process, and demonstrate the practicality of our attack.
- We introduce two new fingerprinting techniques that rely on extensions' communication patterns and are robust against all countermeasures previously proposed by researchers or deployed by browsers.
- We present the largest extension fingerprinting study to date, highlighting the true extent of fingerprintable extensions within the Chrome Store. Our dataset also enables an evaluation of our attacks against a state-of-the-art countermeasure [55], demonstrating the effectiveness of our techniques as Carnus can still detect the vast majority of the behavior-based extensions.
- We present an analysis on the unicity of extensions using publicly available extension reviews as the vantage point for quantifying the uniqueness of extensions among more than 1.16 million users. Apart from measuring the true usefulness of extension fingerprints for tracking users, we explore a novel deanonymization attack where users' identities are inferred based on their public reviews.
- We present the first empirical analysis on the privacy inference attacks enabled by browser extensions. Specifically, we describe an attack for inferring users' personal and sensitive information (e.g., demographics, ethnicity, religion, etc.) based on the intended functionality of detected extensions.

II. Background and Threat Model

Extension fingerprinting. While modern browsers offer a rich set of capabilities and functionality, third-party developers are allowed to create extensions that add new features and enable a better experience for users. For instance, popular extensions can remove undesired content like advertisements [21], [39] but can also harden browsers by removing undesired features [48] or forcing web requests over HTTPS [27], [45]. To achieve their desired functionality, extensions can alter a webpage's DOM and even execute arbitrary scripts in the context of a webpage (which introduces a significant security threat [10], [12], [15], [28], [36]). However, unlike plugins, browsers do not provide a JavaScript call for a webpage to obtain a list of the extensions installed in a user's browser.

As a result, extensions can only be detected by pages indirectly. While we present details on how Carnus achieves this in Section III, the main idea is that extensions expose elements (e.g., an icon) or exhibit behavior that is observable by webpages. If a specific extension's elements or behavior are unique among all extensions, then a page can uniquely identify (i.e., *fingerprint*) it. Identifying exposed resources in Chrome is a straightforward process that has been demonstrated before [24], [47]. On the other hand, uniquely identifying extensions based on their behavior is a challenging task that presents several obstacles in practice. First, extensions can exhibit behavior that is dynamic and potentially ephemeral in nature that also relies on characteristics of the visited website, as opposed to the typically static and long-lasting nature of exposed resources. Moreover, multiple extensions may exhibit similar or identical behavior (e.g., blocking ads on a page). To make matters worse, if a user has multiple extensions installed their behavior may overlap, further obscuring the "signals" used for fingerprinting. While prior work [52] proposed the use of behavioral features for fingerprinting extensions, that study did not actually automatically create or evaluate such fingerprints, nor did it explore the implications of overlapping behaviors from different extensions. In this study we provide

a comprehensive analysis of extension fingerprinting that explores these challenging, yet critical, practical dimensions.

Threat model. In practice, extension fingerprinting techniques can be deployed in different settings, which can affect their accuracy; for instance, certain extensions can only be detected by certain web pages as their functionality gets "triggered" only when the user visits specific domains [52]. Since such extensions cannot be detected by all attackers, we focus on extensions that can be detected regardless of the web page's domain. More specifically, we assume that the attacker is able to lure the user to a specially crafted page that attempts to detect as many installed extensions as possible. Furthermore, as in previous studies, we assume that the user visits the attacker's website over Chrome on a computer and not a smartphone, since the mobile version of Chrome does not support extensions.

III. System Design and Implementation

In this section we provide details on the design and implementation of our system. A high-level overview of Carnus is shown in Figure 1. The first module of our system is responsible for crawling the Chrome Web Store and downloading all available extensions. The crawler also extracts metadata including the descriptions provided by the developers, as well as all accompanying reviews by users. The extensions are processed by both static and dynamic analysis components which identify their WARs and exercise them to extract their behavioral signatures. These are subsequently processed so the final fingerprint is created for each extension. For our privacy inference attacks, we focus on fingerprintable extensions. Indeed, their descriptions, metadata, and users' reviews are processed so as to identify extensions of interest and create the list of user characteristics and traits that they reveal.

Extension enumeration. As mentioned previously, prior studies have demonstrated the feasibility of browser extension enumeration and fingerprintability. These studies focused their efforts on identifying extensions that expose specific resources (i.e., WAR-based enumeration) either directly [24], [47] or with clever implicit approaches [44], [46]. In the following subsections we provide technical details and outline the fingerprint generation and extension detection process for our four techniques. Overall, we present the first study that incorporates multiple fingerprinting techniques, enabling the largest and most comprehensive exploration to date.

A. WAR-Based Extension Enumeration

An extension's structure and required permissions are defined in a *manifest* file. In practice, the permissions declare the extension's ability to access websites and Chrome APIs, and the content scripts point to code that will be fetched and executed in the context of web pages. Resources such as images, JavaScript and CSS files are packed along with extensions. Relevant to our goal are web accessible resources (WAR). The WAR section of the manifest defines a list of resources within an extension that can be accessed from a web page. In other words, a page is only able to access resources whose paths exist in the WAR section [2]. In Chrome, a page can fetch a resource from an extension through: `chrome-extension://<UUID>/<path>`, where `<UUID>` is the public extension ID, and `<path>` is the path to the resource.

By compiling a list of the extensions that expose such resources, a website can probe these resources in order to detect which extensions the user has installed in her browser. Since this attack is only feasible when the extensions' identifiers and resource paths are known, Firefox recently implemented a countermeasure of assigning a random identifier to each installed extension. However, Chrome lacks any countermeasures for preventing WAR-based extension enumeration.

As our goal is to maximize the potential coverage of our attack and explore in depth the privacy implications that arise from the detection of extensions, we implement the WAR-based technique as part of our system. During the preparatory phase, we statically parse the manifest files of the extensions collected by our crawler and identify which ones expose such resources. During the attack phase, a script in our page issues a request for each extension's WAR and determines if the extension is installed based on the status code of the response.

B. Behavior-Based Extension Enumeration

During an initial exploration of the web extension ecosystem, we encountered various extensions that exhibit patterns of potentially detectable behavior. Specifically, we found extensions that dynamically add new images, buttons, or text to the web page, some that detect images and text and replace them, as well as extensions that fetch resources from the web and use message passing for communicating with the JavaScript code inside the visited page. By detecting all the behavioral patterns, a website can generate *behavior-based signatures* that allow identification of the user's installed extensions. In the following we focus on detecting the extensions that alter the DOM tree of the visited web page, while in subsections III-C and III-D we present our new techniques for capturing extensions' intra- and inter-communication patterns.

DOM modification. In general, the types of modifications that are performed by extensions can be attributed to the following behaviors: (i) adding new nodes in the DOM tree of the page, (ii) removing nodes from the DOM tree and (iii) changing the attributes of existing nodes. A special case of the latter category is the case of extensions that identify specific keywords in the text of the page and replace them with other predefined keywords.

To capture the modifications performed by each extension and generate their behavioral fingerprints, we follow a dynamic analysis process where we aim to "trigger" extensions and elicit their functionality. To that end, Carnus incorporates a precisely crafted website under our control (i.e., a *honeysite* with *honeypages*). Specifically, for each extension we launch a new instance of the Chrome browser with only this extension installed and visit our honeysite three times. During these visits we detect the extension's modifications by comparing the content rendered in the browser with the honeysite's original contents and generate the extension's behavioral fingerprints. Our system visits the honeysite three times during the fingerprint generation process, as our empirical analysis showed that this provides a good balance between eliciting different behaviors by the extensions and not significantly increasing the duration of the dynamic analysis.

Since the honeysite is controlled by us (or in the case of an actual attack by the attacker), all the modifications that occur

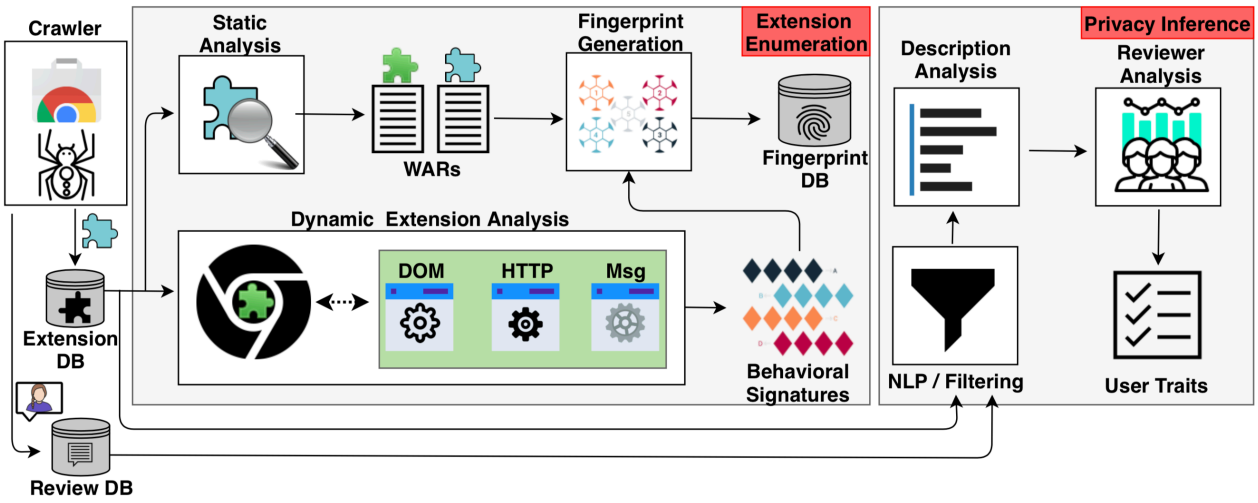

Fig. 1: Overview of Carnus' two main workflows. The extension enumeration phase analyzes extensions and creates signatures that enable fingerprinting browser extensions. The privacy inference phase analyzes extensions and their respective reviews and identifies extensions that implicitly reveal sensitive or personal information about the users.

in the page during the attack phase are the result of the user's installed extensions and not some other external factor, thus, allowing the attacker to isolate precisely the changes performed by extensions. When a user visits our website (i.e., during the extension detection phase), Carnus captures the contents of the website, detects the modifications performed by their installed extensions, and constructs signatures that describe these modifications. Finally, for identifying the user's installed extensions, Carnus matches the visiting user's signatures with the fingerprints that we have previously stored in our database.

Design of honeysite. As the goal is to trigger as many extensions as possible into performing some form of modification, and generate their fingerprint, our honeysite includes highly-diverse content. The main challenge we attempt to tackle is that extensions may exhibit detectable behavior only when specific conditions are met. For example, the popular password manager LastPass inserts an icon in form fields, but may not interfere with any other objects in the page. If the honeysite does not have such a field, LastPass will not insert the specific icon in the page, hence, Carnus will not be able to detect it. To avoid such cases, our honeysite includes all available HTML tags, types, various attributes, ad-fetching scripts (that do not actually fetch any ads) and media resources of various types.

Since the space of all potential extension triggers is vast, including all available HTML tags in the honeysite cannot definitely offer the coverage we aim to obtain, as our system will not be able to detect extensions that are only triggered by specific keywords being present in the page's text. As such, our system tries to identify keywords that need to be included in our honeysite through the following process: we visit each extension's page in the Chrome Web Store twice, once with the respective extension installed and once without, and compare the text of the extension's description across the two visits.[1]

The observation behind this is that, specific keywords that activate such behavior are typically included in the extension's description. Any keywords that are detected when visiting the description page are included in our honeysite. While this is a fairly straightforward approach, it has not been used in prior extension fingerprinting studies and actually enables the detection of 7.6% of all the extensions that we detect through honeypage modifications (22.1% of these also reveal sensitive information in our inference study).

Fingerprint generation. This methodology is followed for the generation of the extensions' behavioral fingerprints that are stored in our database, as well as the signature of the modifications that are performed when the user visits the attacker's website. In both cases, Carnus treats all the observed modifications as a sequence of *additions* and *removals* (replacement or modification of an existing element can be considered as a removal of that element and addition of a new one). We construct the signature by considering all added and removed terms. That is, a signature consists of two distinct parts: (i) the set of additions and (ii) the set of removals. For instance, in the case of an extension that injects a new image in the web page (i.e., `<img src='image.png'>`), the signature will be generated as the following sets `[{"<img", "src='image.png'>"},{}]` that represent the added and removed terms, with the latter one being an empty set in this case. Similarly, for an extension that replaces `image-1` with `image-2`, the signature will be `[{"src='image-2.png'>"}, {"src='image-1.png'>"}]`.

At a high level, after identifying the modifications of all extensions and generating their fingerprints, we can enumerate the extensions of a user by matching the observed modifications' signatures with the extensions' fingerprints that we already have in our database. That is, when a user visits our website, we have embedded JavaScript code that identifies the modifications during that visit, calculates the signatures of these modifications on-the-fly, and compares them with the fingerprints we have previously generated for all extensions.

However, in practice, there are two important challenges that can significantly affect the behavior-based detection of extensions and lead to false positives or negatives. First, exten-

[1] Descriptions can be found at https://chrome.google.com/webstore/detail/UUID

sions can exhibit different behaviors across different executions or inject content that contains dynamic parts. Second, multiple extensions may perform similar modifications on the website's DOM tree, which can affect the accuracy of our system. Next, we describe the process we have established for solving these issues and making Carnus more robust.

Dynamic content. As mentioned before, during the fingerprint generation phase we visit the honeysite three times. This allows us to differentiate between extensions that always perform the same modifications and those that exhibit different but likely similar behaviors. If these visits generate different behavioral fingerprints, we keep them all in our database as the extension's fingerprints. We estimate how similar or different these fingerprints are and detect whether some parts of them include dynamic content. An example of fingerprints that change with every execution is given by the following:

```
[{"<img","src='img.png'>","timestamp=100"},{}]
[{"<img","src='img.png'>","timestamp=200"},{}]
[{"<img","src='img.png'>","timestamp=300"},{}]
```

In the case where the extension injects dynamic content, the user's signature will never directly match the extension's fingerprint that we have already generated and stored. To handle such cases, during the fingerprint generation phase Carnus tries to identify the static and dynamic parts of highly similar fingerprints (i.e., that have all but one terms identical, and a single term *partially* matching) and re-writes them so that the dynamic part of the partially matching term is not included in the fingerprint. In the above example, Carnus will include the matching part (i.e., "`timestamp=`"), but it will omit the value that follows the "=" sign.

The approach we described above for the detection and omission of fingerprints' dynamic values is a bit conservative, as it only considers the case of almost identical fingerprints that have all their components matching or partially matching. Since this approach cannot detect all cases of fingerprints with dynamic parts, we also allow a certain number of components to *mismatch* when comparing the fingerprints in the database with the visiting user's signature. The number of allowed mismatches is determined according to the size of the fingerprints (i.e., number of terms in the sets of *additions* and *removals*). Since smaller fingerprints tend to be more specific and also have a higher risk of a false positive matching, we enforce a strict policy of no mismatches allowed for fingerprints that have a size of up to 10 (covering almost 55% of our extensions as shown in Figure 4). For larger fingerprints, with a size of 10 to 50, which covers an additional ~26% of the extensions, Carnus is more lax and allows mismatches of up to 10% of the fingerprint's size. For the final ~20% of even larger fingerprints we allow mismatches of up to 5% of the fingerprint's size.

Fingerprints overlap. When comparing the extensions' fingerprints that are stored in our database with the visiting user's signature, we essentially compare the two sets of added and removed terms of every stored extension's fingerprint with the respective sets of added and removed terms in the user's signature. To have a match both sets of a fingerprint need to match those of the user's or a subset of them (i.e., the user has multiple extensions installed and her signature consists of the modifications performed by all of them). However, since there are extensions that perform similar modifications, it's possible to end up with overlapping fingerprints. In such cases, the fingerprint of an extension appears to be the same or part of another extension's fingerprint. As this can result in false positives (all overlapping fingerprints will match the user's signature), after detecting all the matching fingerprints, we try to identify and resolve such cases.

In the case where two identical fingerprints match the user's signature, our system cannot determine which one of the extensions the user has installed. Therefore, we consider both of the extensions unless one of them can be matched by another technique of our system (i.e., WAR-based or communication-based). When one of the matched fingerprints appears to be a subset of another matched fingerprint, Carnus keeps the one that has the highest number of terms matching the signature.

C. Intra-communication Based Enumeration

For security reasons, browsers separate the execution context of extensions' background scripts, content scripts, and the page's scripts. These scripts run in isolated worlds, preventing one from accessing variables and functions of the others [1]. However, they can communicate by exchanging messages [3]. Content scripts can communicate with background scripts by using the `runtime.sendMessage` API. Background scripts can use the `tabs.sendMessage` API to communicate with content scripts. The messages exchanged between the extensions' background and content scripts are invisible to the page.

Furthermore, communication between an extension and a web page can be achieved in two ways: the page's scripts can exchange messages with (i) the extension's background scripts and (ii) content scripts. For the first approach, a page can use the `runtime.sendMessage` API to send messages to the extension's background, and the extension in turn uses the `runtime.onMessageExternal.addListener` API to receive these messages and send responses back to the page. However, this communication is only possible when the extension adds an `externally_connectable` key in its manifest file, specifying the URL patterns of websites that the extension wants to communicate with. The URL pattern must contain at least a second-level domain, and wildcard style patterns like "*" or "*.com" are prohibited. This is to prevent arbitrary websites from communicating with the extension.

For communication between an extension's content script and a web page, the `postMessage` API can be used (and the `externally_connectable` key is not required). As a result any arbitrary web page can exchange messages with the extension. In this section, we leverage this kind of message-passing to create a new extension fingerprinting vector. Differences in the messages sent by extensions allow Carnus to distinguish between different extensions that employ message passing for intra-communication purposes. For instance, Listing 1 shows parts of the content script of the "MeetMe Dolby Voice 1.1" extension (UUID: `lflnplggpolkcgknahacafilopgngelc`), which sends two messages to the web page.

Fingerprint generation. The approach that we follow for capturing such messages is similar to the one we implemented for detecting DOM modifications and generating behavioral fingerprints. We include a JavaScript `EventListener` in our honeysite to capture and log all `message` events. Again, we visit the honeysite three times for each extension to identify

```
function logToJavascriptPlugin (msg) {
window.postMessage({MeetMeDolbyVoiceMsgP1x1:
    'log_msg', raw_value:{component:'ChromeExt-FG',
        message: msg}},'*');
}
...
logToJavascriptPlugin('Sending \'ping\' message to
    transport layer');
window.postMessage({MeetMeDolbyVoiceMsgP1x1:'ping'},
    '*');
```

Listing 1: Code snippets of an extension that sends two messages to the web page.

```
[{"MeetMeDolbyVoiceMsgP1x1":"log_msg", "raw_value":{
    "component":"ChromeExt-FG", "message":"Sending '
    ping' message to transport layer"}},
{"MeetMeDolbyVoiceMsgP1x1":"ping"}]
```

Listing 2: Example of an intra-communication fingerprint.

whether it always sends the same messages and if they contain any dynamic values. After removing the dynamic parts, the set of exchanged messages is used for generating the extension's fingerprint. Listing 2 shows the fingerprint that is generated for the aforementioned extension (that was presented in Listing 1).

Extension enumeration. During the attack phase, when a user visits our website, our system captures all the messages sent by the installed extensions and matches them with the message-based fingerprints that we created during the message capturing phase. To capture the exchanged messages, as described above, we include an `EventListener` in our website and log all received messages. After constructing the user's message-based signature, Carnus checks which of the extensions' fingerprints are a subset of it, indicating that those extensions are installed in the user's browser. For this enumeration technique, 20% mismatches are allowed. Finally, from the list of detected extensions with this approach, we remove extensions if their fingerprint is a subset of a fingerprint of another detected extension.

D. Inter-communication Based Enumeration

Extensions can issue HTTP requests for fetching resources (i.e., css files, scripts, images, etc.) from the Internet. For instance, the HTTP requests that are issued by the "source now" extension (UUID: `dimnlaemmkbhojonandnnbogfifjnpno`) are shown in Listing 3. Carnus incorporates a novel extension-detection module that relies on monitoring all the HTTP requests issued by extensions for fetching resources.

For detecting HTTP requests issued by the user's installed extensions, we use the Resource Timing API [57], which stores performance metrics regarding the performance and execution of web applications and is accessible through JavaScript. Using the `performance.getEntriesByType("resource")` method we can query the list of all resources requested. As a result, we obtain all resources requested by the web page and content scripts of extensions installed in the user's browser.[2] Such requests can exhibit unique features, thus rendering them a useful signal for enumerating installed extensions.

[2]Resources requested by extensions' background pages are not included.

```
[{"https://b.alicdn.com/@sc/list-buyer/assets/source
    -now/entry/index.js"},
{"https://b.alicdn.com/@sc/list-buyer/lib/js/jquery.
    js"}]
```

Listing 3: Example of an inter-communication fingerprint.

Fingerprint generation. As before, we visit our specially crafted honeysite and detect and record the URLs of all requested resources. Since, in practice, the attacker creates and controls the honeysite, it is trivial to detect any issued requests that are not part of the page but originate from extensions. During our dynamic analysis we visit our honeysite three times to detect whether an extension always fetches the same or different resources, and accordingly generate the extension's fingerprint based on the set of these URLs.

Extension enumeration. When a user visits our website, we capture all the outgoing HTTP requests in the same fashion and determine which requests appear due to the installed extensions. Thus, we generate the signature of the visiting user as the set of these requests, and try to match the extensions' fingerprints that we created previously with the user's signature. As with the intra-communication technique, we allow 20% mismatches and remove any detected extensions that have a fingerprint that is a subset of another detected extension.

Overall, we follow different mismatch thresholds for the DOM-based and communication-based fingerprints. Since some DOM-based modifications are common across different extensions, and the extensions' behavior and fingerprint size vary significantly, we found that an adaptive approach based on the fingerprint size is more effective. For communication-based fingerprints, which are significantly smaller than the DOM-based ones, as well as more unique and robust, we empirically found that a lax heuristic of allowing 20% mismatches yields better results.

E. Behavior-based Fingerprinting: Current State of Affairs

Prior work. Starov and Nikiforakis [52] proposed the method of detecting DOM-based modifications for fingerprinting the user's installed extensions and presented XHound, a tool for identifying if an extension is detectable based on the modifications it performs on the page's DOM. A followup study by Trickel et al. [55] also leveraged some functionality of XHound. While these two studies refer to behavior-based fingerprinting, they *did not* actually create any behavior-based fingerprints automatically or provide technical details on how these fingerprints can actually be created. In more detail, when discussing their implementation of checks (that compare DOM changes to signatures) for detecting extensions, the authors of XHound explicitly state that "these checks could, in principle, be automatically generated by parsing XHound's output but we leave this automation for future work" [52]. In [55] the authors *manually* created behavior-based fingerprints for 20 extensions to evaluate their proposed countermeasure, and stated that behavior-based fingerprinting "does not currently scale" and that fingerprint creation "requires human intelligence and no recent research has shown how to automatically generate".

While XHound's proposal of using DOM-based changes for fingerprinting is a major contribution, it is important to

```
1.[{style="display:','none;"','id="hashmenu01"'},{}]

2.[{'class="rmcScreenshotInstalled"'},{}]
```

Listing 4: Example of behavioral fingerprints that are not effective against the countermeasures of CloakX.

```
1. [{'src="//buy.dayanghang.net/inject/common.js"'},
    {}]

2. [{'action="/cconcert-login"','style=""'},{'action
    ="/cpanel-login"'}]

3. [{'value="mata-inactive-38.png"','id="mata-icon-
    name"','type="hidden"'},{}]
//This fingerprint will be rewritten to [{'value="
    mata-inactive-38.png"', 'type="hidden"'},{}]

4. [{'type="text/javascript"','src="chrome-extension
    ://nogempgplicnckhcmgjjjgflmipmbgaf/variables-
    sharing.js"'},{}]
//This fingerprint will be rewritten to [{'type="
    text/javascript"','src="chrome-extension://
    nogempgplicnckhcmgjjjgflmipmbgaf/"'},{}]
```

Listing 5: Example of behavioral fingerprints that remain effective even after the deployment of CloakX.

highlight all the challenges posed by the full process that our system needs to address; this includes automatically identifying all the changes in the DOM, generating the signatures for a given extension, comparing those to the signatures of other extensions and removing redundant overlapping parts, evaluating how extensions co-interfere in practice, as well as optimizing the system to complete the attack in a short time. Our research fills this significant gap by providing technical details on how to create behavior-based fingerprints, demonstrating the *automated* creation and detection of such fingerprints *at scale*, and exploring their effectiveness in practical settings.

Countermeasures. Trickel et al [55] proposed CloakX, a system that aims to render extension enumeration ineffective by diversifying the attributes of fingerprints. To prevent extension detectability, ClockX substitutes the values of `ID` and `class` attributes of the injected DOM nodes with randomized values. In addition to that, it also injects random tags and attributes in the page. It does so to: (i) inhibit websites that use DOM queries (i.e., methods `getElementsByClassName()`, `getElementsByTagName()` and `getElementById()`) from identifying specific elements that are injected by an extension and (ii) make structural patterns noisy.

In Listing 4 we present two examples of Carnus fingerprints that are rendered ineffective by the countermeasures of CloakX [55]. Since CloakX substitutes the values of the `ID` and `class` attributes, if we exclude these attributes from our fingerprints, the first fingerprint in Listing 4 becomes too generic after excluding the `ID` attribute, while the second becomes empty after removing the `class` attribute.

However, our approach does not simply rely on the identification of specific elements and tags that are injected, but analyzes all the changes in the honeysite, *term-by-term*, to construct the fingerprints of each extension. Our system considers all terms added and removed by each extension and, by design, can detect and filter out *noisy terms* that could make the fingerprints unstable. Listing 5 presents examples of Carnus fingerprints that remain effective even if a user's browser relies on CloakX for protection. The first two fingerprints are not changed at all, while the following two are modified but still contain unique terms that lead to their identification. Moreover, CloakX does not alter extensions' intra-browser and external communication patterns. We further explore our attack's effectiveness against CloakX in Section V.

IV. Extension-based Inference Attacks

While the set of extensions that are detected by our enumeration techniques can be used as a vector for fingerprinting, these extensions can also reveal previously unknown and potentially sensitive information about the user. This includes her personal traits and interests, religious and political beliefs, demographics, etc. In this section, we present the techniques employed by Carnus for analyzing extensions, understanding the functionalities they perform, and finally, extracting interesting and potentially sensitive information about these users.

This analysis uses as input the extensions' descriptions and reviews that our crawler collected from the Chrome Web Store. Since it is inherently hard, if not impossible, for someone other than Google to collect information about all the users that have installed each extension, we use users' public reviews as a substitute; This provides a unique view within the extension ecosystem, allowing for an aggregate analysis on a very large number of users. Moreover, users are required to install an extension before being allowed to provide a review, resulting in a dataset of users that have *actually* installed the extensions.

Topic classification. The first phase of our analysis lies in understanding what functionality is offered by each extension and then classifying them into distinct categories according to their type. For this task we use the extensions' descriptions. At first, we pre-process and "clean" the noisy description text so as to remove "irrelevant" text that can affect the outcome of the classification (more details below). For the actual classification we use Google's Natural Language API [4], which is highly accurate as we outline in Section VI. Google's API provides 35 categories and 400 subcategories. We manually identified and grouped all the categories/subcategories that refer to the same (or related) topic under a generic label, so as to provide a more concise categorization of the information that is pertinent to our analysis. For example, we group together all health-related categories (i.e., Health Conditions, Neurological Conditions, etc.), under the label "Health".

Pre-processing and cleaning descriptions. Typically each extension is supposed to have two descriptions, a short one that describes the extension's functionality in one or two sentences, and a longer one that provides more details about its functionality, implemented features, supported websites, etc. In practice, however, some of the extensions omit one or both of the descriptions, or contain text that is not useful for classification. The main challenge is identifying which parts of their text are relevant and useful for classification, and which have information that could potentially lead to incorrect results and needs to be removed. This task is challenging since there are no enforced guidelines regarding the content that should be included in descriptions.

We start our processing by detecting the language of each description and translating non-English ones into English. Text is then split into paragraphs and sentences, and our system tries to detect and remove text that corresponds to very short or improperly composed sentences. Carnus incorporates the NLTK library [42] for part-of-speech tagging and removes sentences that do not contain at least one noun and one verb. Finally, our system uses NLTK's implementation of the TextTilting algorithm [25] to segment the text into multi-paragraph topical blocks and, since the extensions' functionality is more likely to be described at the beginning of the description, it extracts the first two such blocks for classification.

Description-based inference. We explore whether the extensions' descriptions reveal any sensitive information about the users, such as their location, language, political inclination or religious beliefs. For this task, we use the spaCy library [20] to detect entities that correspond to locations (i.e., countries, cities), nationalities, languages and ethnic, political or religious groups. With this approach we detect and verify that 2,260 of the extensions indeed reveal such information about their users. We note though that this number corresponds to a lower bound as it depends on the library's accuracy in detecting such information, and improving named entity recognition techniques is out of the scope of this work.

During the verification we only consider extensions for which our process extracts specific demographic information, and not cases where additional knowledge is needed for inferring such information. For example, we do not consider entities that correspond to organizations or companies. Although such entities could be associated with particular countries, the task of identifying these associations would require significant manual effort. Furthermore, we use public wordlists of religious and medical terms [17] and create a new wordlist containing political terms, to detect extensions with descriptions that reveal these types of sensitive information.

Reviewer-based inference attacks. Apart from the description text, we also utilize users' reviews for each extension to extract information that enables the inference of sensitive user data. First, we explore the feasibility of inferring a user's ethnicity based on a fingerprinted extension. Specifically, we are motivated by the observation that the users of a certain extension can exhibit strong ethnic affinity due to its intended functionality (e.g., an extension for buying subway tickets in a specific region). While actual demographic information about the extensions' users is obviously not available, the majority of reviews include the user's name.

In practice, users can only review an extension in the Chrome Store *after* they have installed it in their browser. More importantly, the reviewers' names that appear in the reviews are actually the names from the users' Google accounts and not an arbitrary input provided by users at the time of the review. Furthermore, while in the past reviews could either be anonymous or include the user's name, the anonymous option is no longer allowed (in our dataset we find that only 5.25% of the reviews are anonymous). While this lends considerable veracity to our dataset, the inferred information can be misleading if a Google account's name is a pseudonym. Reviews with nicknames or fictional names that do not match entries in our name lists (described below) are discarded, but fake accounts that use actual names will still be used in our analysis. This is a limitation of our approach as there is no straightforward method for detecting whether the name used during the creation of a user's Google account is fake or not (e.g., consider a simple scenario where someone named "Jack Smith" creates a Google account under the name "John Doe").

TABLE I: Number of extensions detected by each technique employed by Carnus, including those that are unique to each technique (i.e., cannot be detected by any other technique).

Detection technique	**Detected Extensions** Total	Unique
WAR-based	25,866	23,046
Behavioral (DOM-based)	5,793	2,998
Inter-communication	859	181
Intra-communication	450	105

Our analysis shows that while certain names do not reveal much information as they can be commonly found in several countries (e.g., John) others can provide a strong indication of the user's ethnicity. To that end, we use an extensive set of name-by-origin lists (all names per country/ethnic group) created from online resources. We correlate the reviewers' names to the lists to identify the countries or ethnic groups where this name appears, and construct a vector of associated ethnicities (e.g., the name "Deepika" is predominantly found in India and Nepal). As aforementioned, fake names or pseudonyms that do not match a known name from our list are discarded. By combining the ethnicity vectors of the users that have installed a specific extension, we create a breakdown of the demographic information of each extension's user population.

Extensions can attract a wildly diverse set of users and, thus, not all of them are useful for inferring a targeted user's characteristics. As such, we need a method to filter out such extensions and focus on those that have a more consistent user profile. To identify suitable extensions, we use the Shannon-Wiener index [49], which is commonly used in ecological and biological studies for calculating the richness and diversity of a given species, to pinpoint extensions with predominant user ethnicities. An important dimension of the privacy invasiveness of this approach is that it can bypass common technical measures taken by users to hide their country of origin or ethnicity (i.e., VPNs and proxies). Finally, we map users' names to their gender, and discard ambiguous names associated with both male and female. Here we only need to set a prevalence threshold for deciding which extensions are useful for this type of inference; e.g., depending on the scenario, an attacker might consider any extension where one gender accounts for at least 80% of the users as sufficient confidence.

V. Experimental Evaluation: Fingerprints

Extension enumeration. We run Carnus on a collection of 102,482 extensions and find that 29,428 unique extensions can be identified by our system. We find that WAR-based fingerprints can detect 25,866, of which 23,046 cannot be identified through other techniques. Behavior-based fingerprints are the next most effective approach with 5,793 detections, of which 2,998 are not detectable otherwise. Through inter-communication patterns we detect 859 extensions and through

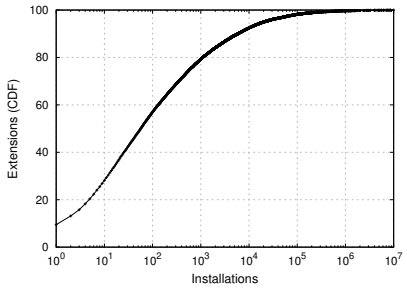

Fig. 2: Number of installations for all the extensions in our dataset.

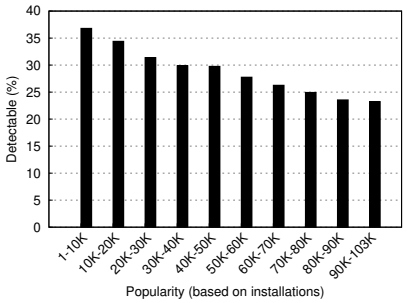

Fig. 3: Correlation of detectability of extensions and their relative popularity.

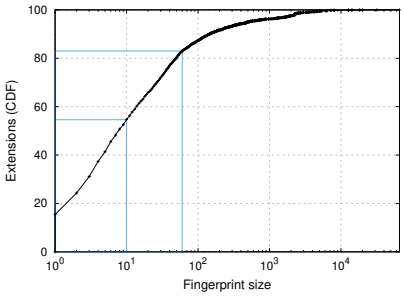

Fig. 4: Extensions' behavioral fingerprint sizes. The blue lines denote where our mismatch thresholds change.

TABLE II: Comparison to previous studies.

Paper	Attack	Platform	Extensions	Detectable
[52]	Behavioral (DOM)	Chrome	10,000	920*
[47]	WAR	Chrome	43,429	12,154
		Firefox	14,896	1,003
[24]	WAR	Chrome	13,000	5,107
[44]	WAR side-channel	Chrome	10,620	10,620
		Firefox	10,620	10,620
[46]	WAR revelation	Chrome	10,459	1,932**
		Firefox	8,646	1,379
Ours	Multi-class	Chrome	102,482	29,428

Note: We convert to absolute numbers when the original work reports percentages. *Estimation based on detectable changes to DOM tree; signatures were not created or tested. **Number estimated by authors since the presented attack relies on random UUIDs which have not been deployed by Chrome yet.

intra-communication we detect 450; the number of extensions that cannot be detected by other means is 181 and 105 respectively. The number of extensions that can be detected by each one of the four techniques are summarized in Table I. While our two new communication-based techniques detect a smaller number of extensions, this is because such behavior is less common among extensions and not due to limitations of our fingerprint-generation process. Furthermore, these two techniques are able to detect extensions that are not detectable by previously known techniques.

Next, in Table II we compare to prior work and find that our system has created the largest set of detectable extensions to date. Most of these studies [24], [44], [46], [47] focused on detecting extensions through exposed WARs (either directly or through some indirect/side channel) – we statically analyzed over 102K extensions to create the most complete collection of Chrome extension WAR fingerprints. More importantly, we create the first collection of automatically generated behavioral extension fingerprints, which enable our novel analysis and evaluation of their deployment in a realistic setting.

Figure 2 presents the number of installations for the 102,482 extensions in our collection. Around 43% of the extensions have more than 100 installations, while around 20% of them have been installed more than 1000 times. In Figure 3 we compare the detectability of extensions with their popularity. We calculate their relative popularity based on the number of installations of each extension and find that there is a clear correlation, as more popular extensions have a higher likelihood of being detectable by Carnus.

Furthermore, for the extensions that modify the page's contents, we find that 5,119 out of the 5,793 (88.36%) extensions always perform the same modifications (i.e., they have a single behavioral fingerprint). For the extensions that exhibit more than one behaviors, we find that 177 (3.05%) have two different fingerprints (i.e., the three runs produce two identical fingerprints and one that is different from the other two) and 497 (8.57%) extensions that have three different fingerprints. Figure 4 presents the size of the behavioral fingerprints of the extensions in our dataset. For the extensions that have more than one fingerprints, in Figure 4 we consider the extension's fingerprint with the largest size. We find that more than half of the extensions (54.6%) have a small fingerprint (up to 10 terms), revealing that extensions typically do not heavily modify pages. Around 26% have fingerprints of 10 to 50 terms, while 19.5% have fingerprints larger than 50. Finally, less than 4% of the fingerprints contain more than 1K terms; these extensions inject entire scripts, like extension UUID: `ohahllgiabjaoigichmmfljhkcfikeof`, or CSS files, like UUID: `ngkinkknobojamikjhodnojnpkbgpddp`.

Practical extension enumeration. While detecting a standalone extension is a fairly straightforward task, we are also interested in evaluating our system's extension enumeration capabilities when multiple extensions are simultaneously installed, as would be the case in a realistic scenario. We setup an experiment where our system randomly selects and installs K fingerprintable extensions from our dataset and visits our honeysite. We only use fingerprintable extensions since non-fingerprintable extensions do not affect detection or interfere with other extensions in any way, and using them would artificially boost our true positive rate. As such, this experiment truly explores the challenge of extension enumeration in practical settings, and is the first to shed light on the intricacies of behavioral-based extension fingerprinting.

Table III presents the results of this experiment; we calculate scores over 100 independent runs for each size of K, where in each run K extensions are randomly installed. TP refers to correctly detected extensions, FP denotes extensions incorrectly detected as installed, and FN is installed extensions that our system could not detect. Since Carnus can detect more extensions than those that are actually installed, the TP+FP

TABLE III: Carnus' accuracy in multi-extension environments.

	2	3	4	5	6	7	8	9	10
TP (%)	97.5	97	98	98.6	98.5	97.6	98.9	97.5	98.9
FP (%)	0.5	4	7.25	5.4	6.2	6.7	3.4	7	2.5
FN (%)	2.5	3	2	1.4	1.5	2.4	1.1	2.4	1.1
F1 (%)	98.5	96.5	95.5	96.7	96.3	95.5	97.8	95.4	98.2

percentages can add up to more than 100%, e.g., if the user has 4 extensions installed but our system returns 5 detected extensions. An important detail is that certain extensions have the same functionality, perform the same modifications and have identical fingerprints. This can occur because developers publish multiple instantiations of the same extension (e.g., in different languages). For example, the extensions "TinyFilter PRO", "Tiny WebFilter" and "WebFilter FREE" are offered by the same developer and have the same functionality. Similarly, extensions like ad-blockers exhibit essentially the same functionality can be indistinguishable. We find that 349 extensions are affected by such ambiguous fingerprints, which is less than 5.5% of the extensions that are fingerprintable through our behavioral techniques. In the table we do not count the additional labels of extensions with identical fingerprints as false positives. For instance, in the aforementioned example, the three identical extensions will be considered as *one* label when calculating the FP rate.

As shown in Table III, our system correctly identifies ∼97-99% of the installed extensions in all cases, indicating the consistent accuracy of our system. The extensions that Carnus misses (i.e., FN: ∼1-3%) are extensions that perform new modifications for which we do not have a fingerprint or are the result of extension co-interference. After analyzing our results we found that the main reason behind these false negatives is the co-interference between the installed extensions, where the modifications of one extension can affect the modifications of the other. This co-interference can also cause false positives, as the combined effect of multiple extensions can result in matching the fingerprint of an extension that is not installed in the user's browser. Another reason for false positives is that Carnus allows certain mismatches when comparing fingerprints, which can lead to misclassifying extensions that have similar fingerprints and whose differences fall within the range of allowed mismatches. The FP rate is less consistent, with an average of 4.77% across all values of K. If we do include the labels of multiple identical extensions as false positives (e.g., in the previous example 2 of the 3 identical extensions would count towards the false positives) our average FP rate across all sizes of K becomes 8.1%. Nevertheless, despite the challenging nature of behavior-based fingerprinting in practice, our system is highly accurate with an F1 score of 95.4-98.5%.

Countermeasure effects. Trickel et al. [55] recently proposed CloakX as a defense against extension fingerprinting. While their approach is obviously not effective against our inter- and intra-communication fingerprints, we want to quantify its effectiveness against our other behavior-based fingerprints that fall within their threat model. In that work, they separate behavior-based fingerprints into two different categories, namely *anchorprints* and *structureprints*. However, since our behavior-based fingerprints cover both of their categories, for ease of presentation we will continue to refer to them as behavior-based fingerprints. We refer the reader to their paper for the full details behind their proposed countermeasure but, in a nutshell, their system randomizes the values of `ID` and `class` attributes to prevent behavior-based detection. They also inject random tags, attributes, and custom attributes into each page, and randomize the *path* of *web-accessible resources*. As such, we analyze our behavioral fingerprints and quantify the effect of their proposed countermeasure on our system.

Since CloakX randomizes `ID` and `class` attributes, we first quantify the effect of removing all such `ID` and `class` element additions from the behavioral fingerprints. We find that the fingerprints of 2,790 (48.16%) extensions do not rely on such elements and are thus not affected by the proposed defense. Out of the remaining fingerprints, we find that 751 (12.96%) are affected in a way that would prevent uniquely identifying the extensions. When we also consider our communication-based fingerprints, 51 of these 751 extensions can be identified. Thus, 5,093 (87.92%) extensions are not affected by this countermeasure.

To prevent WAR-based detection CloakX replaces extensions' WAR paths with a randomized value. While this countermeasure is effective against WAR-based detection, it does not affect our behavior-based detection. When a WAR URL (i.e., `chrome-extension://<UUID>/<path>`) is included in the extension's behavioral fingerprint, CloakX can only randomize the resource's path and not the UUID. Thus, we can discard the randomized path from the behavioral fingerprint, as shown in the last example in Listing 5, and the fingerprint will still be unique among all our behavioral fingerprints.

Regarding the effect of tags and attributes being randomly added by CloakX, this can be counteracted using Carnus's mechanism for detecting and removing dynamic content from the fingerprints. Specifically, by visiting our honeysite multiple times during the fingerprint generation, our system can detect which *added terms* remain the same across visits and which ones change. For the extensions that have only one behavioral fingerprint in our database[3] Carnus can safely filter out the randomly added artifacts, without affecting the extensions' fingerprints. To that end, from the 5,093 extensions that our system can identify after removing the fingerprints with `ID` and `class` attributes, we end up with 4,800 extensions that have only one fingerprint in our database (313 of them were re-written to remove dynamic parts of partially matching terms). From the 293 extensions that have fingerprints that could potentially be affected by CloakX randomly adding tags and attributes, 250 do not have any communication-based fingerprints. Even though the random tags added by CloakX to the fingerprints of these 250 extensions can most likely be identified and removed with a sufficient number of visits to our honeysite, in the *worst case* scenario where our system is not able to remove the added tags and attributes for any of these 250 extensions, 4,843 out of the 5,793 (83.6%) extensions that have behavioral fingerprints will remain unaffected. Overall, Carnus will be able to uniquely identify 83.6% - 87.92% of the extensions that have behavioral fingerprints even if CloakX is deployed.

[3]This includes extensions that always perform the same modifications, and extensions with fingerprints that differ only because of *partially matching terms*, which our system re-writes into a single fingerprint after discarding the dynamic part of the partially matching terms, as explained in Section III-B.

System performance. As discussed in Section III, during the fingerprint generation phase our system visits our honeysite with a single extension installed and captures all the modifications, message exchanges, and resource fetching conducted by the extension. Carnus waits for 15 seconds before capturing the contents of the page and generating the behavioral fingerprints, so as to allow enough time for all the modifications to take place and the external resources to be fetched. The processing for generating the fingerprints (i.e., constructing the sets of *added* and *removed* terms) takes less than 1 second. Since each extension needs to be dynamically analyzed 3 times, we parallelize 3 different browser instances and the overall time that is spent for exercising each extension during the dynamic analysis phase does not exceed 16 seconds. This process is performed once per extension and only repeated if a newer version of an extension is released; given the low overhead it is more than suitable for practical large scale deployment.

A more critical dimension of a fingerprinting system's performance is the time required for the extension detection phase. During our implementation, our goal was to minimize the overhead that our system imposes on the client side and, thus, minimize the time a user needs to stay on our website for Carnus to detect her installed extension. For this, we offload all the processing for behavior-based detection to the server, which includes matching the modification and communication signatures with the stored extensions' fingerprints etc. The JavaScript code in our page that is responsible for the WAR-based detection, obviously, needs to run on the client side.

To assess this aspect of our performance, we conduct experiments using an off-the-shelf commodity desktop machine with a Quad Core Intel i7-7700 and 32GB of RAM. Specifically, we automate a browser instance that has 4 extensions installed to visit our honeypage. We then measure the time that is required for processing to complete both on the client and server side. We run this experiment 300 times with a different set of 4 randomly-selected extensions installed each time. The client side processing requires 8.77 seconds on average (*stdev*: 0.39), with a median value of 8.58 seconds. The server only requires 3.62 seconds on average (*stdev*: 1.83), with a median of 2.94 seconds. In other words, since the backend processing is not dependent on the user remaining on the page, Carnus requires the user to stay on our honeysite for less than 10 seconds to successfully detect the installed extensions. This highlights the efficiency of our attack and its practicality in deployment in realistic scenarios. To examine whether the number of installed extensions affects the processing time that is required, we repeat the experiment with 5 extensions being installed, and find that the average duration remains essentially the same.

VI. Experimental Evaluation: Inference

While extension enumeration can be used as part of the browser fingerprinting process, the set of detected extensions can also be used to infer sensitive information about that user, which could enable or facilitate a wide range of privacy-invasive practices, from government surveillance of religious minorities [7] to tailored advertising that targets sensitive topics [16], [34] (e.g., health issues).

Extension classification. The first phase of our inference attack uses Google's Natural Language API for identifying the categories that better describe each extension. This allows us to classify 20,409 of the extensions in our dataset; the remaining 9,019 extensions could not be assigned to any category, mainly due to them having a very short description text.

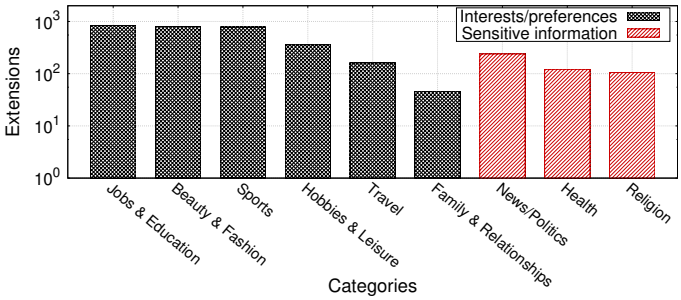

Fig. 5: Categories of extensions that reveal personal and potentially sensitive information.

As one might expect, the most popular category is that of *Computing* (subclasses: *Multimedia*, *Programming*, *Internet Software*, etc.) with 7,652 extensions. The next most popular category is related to *Social Networks* with 4,977. While such categories do not reveal any information that is interesting from a privacy perspective, there are, however, other categories that reveal more personal information. In Figure 5 we present the main such categories and distinguish between those that reveal important but non-sensitive information (e.g., the user's interests) and those that reveal sensitive information such as a user's health conditions, religion and political views.

For instance, in the *Health* category we can find extensions such as UUID: `knijgomkfcdigmbogcnffcbfgmapmkca`, which is designed to assist people with dyslexia, and UUID: `edmpbiamnlgdichailkadlagmbklhbpk`, which allows users to compare their own images to visually similar skin cancer images on the web. In the *Religion* category there are extensions like UUID: `ndmbeogingkjkmmkoomnigifmpajmbkc` and UUID: `apkkllhahggfkhfdlmpopcncnaiclfdm`, which expose the user's religion.

Our classification results in assigning 838 extensions to the *Jobs & Education* category, which is the most prominent one, and 46 to the *Family & Relationships* category, which is the least common one among the less-sensitive categories. For the most sensitive categories of *News/Politics*, *Health* and *Religion* our classification results in 238, 121 and 105, respectively (the list of extensions in these categories is available online [5]). To assess the accuracy of the classification we randomly chose 50 extensions from each one of the three sensitive categories, and manually checked if they were assigned to the correct category or not. Through this manual process we found the accuracy of the classification to be 100%, 86% and 80% for the respective categories of *News/Politics*, *Health* and *Religion*.

Ethnicity inference. Next we analyze our fingerprintable extensions and calculate the Shannon-Weiner index (SWI) of the ethnicities inferred based on the names of the reviewers. Since this index incorporates both the richness and evenness of the population (i.e., reviewers' ethnicities), we found that for extensions with a fair number of reviews, a threshold of 3.5 is sufficient to indicate whether an ethnicity is prevalent; in practice attackers can fine-tune this threshold based on their

requirements. Our analysis shows that this technique can identify 12,754 (43.33%) extensions with a prevailing ethnicity. To further increase our confidence, if we only consider extensions that have been installed by at least 500 users and have reviews by at least 20 different users, 2,593 extensions can be used for this type of inference.

As this approach is topic agnostic, i.e., does *not* rely on the extensions' description or type functionality, it enables the inference of information that is well hidden and, practically, unavailable. For example, the "FlashSaleTricks" extension (UUID: `bboalniaekhennojedffbbjlokcpbjgn`) has a Shannon-Weiner index of 2.62. The language of that extension, and the text of its description, is English, but Indian names are predominant in its reviews. By checking its website (https://www.flashsaletricks.com/) we found that it indeed targets Indian users. An interesting case is that of the "Download Master" (UUID: `dljdacfojgikogldjffnkdcielnklkce`), which appears to be popular among Russian users (SWI=3.47). While this extension is in English, we found that upon installation it downloads additional software that is in Russian. Similarly, while the description of the "J2TEAM Security" extension (UUID: `hmlcjjclebjnfohgmgikjfnbmfkigocc`) is in English, the majority of its reviewers are Vietnamese (SWI=3.21). In another example, the "wanteeed" extension (UUID: `emnoomldgleagdjapdeckpmebokijail`), with an index of 3.29, is shopping-related and is popular with French predominantly female users (2.9x more female names).

Sensitive information inference. To further understand what type of information can be inferred from the presence of specific extensions, and which extensions reveal such information in practice, we investigate whether the languages of an extension can be used for characterizing the user. To that end, we collected the languages that are supported by each extension from the Chrome Web Store. We find that 24,392 (82.88%) of the extensions only support a single language, and that 5,425 of them (18.43% of detectable extensions) are in a language different from English while 4,623 (15.7%) have English (United States) as their language. Moreover, for extensions that support multiple languages we find that 1,747 out of 4,983 support 4 or less languages. Extensions with an extensive list of languages cannot, in practice, provide any insights about the user. Finally, apart from the extensions' languages that are listed in the Web Store, 3,922 (13.32%) extensions have a description in a language other than English, which indicates that those extensions target a specific language-speaking population. While extensions that are exclusively in English cannot be used for determining the origin of the user, most of the other languages can provide strong indications about the user's ethnicity, country or residence/origin. Our analysis identified a total of 7,552 (25.66%) non-English extensions that reveal the language of the user.

To further explore what sensitive information can be inferred from extensions, we conduct a more in-depth analysis on the extensions' description text. First, we use spaCy's Named Entity Recognition (NER) [20] to identify entities in the extensions' description that expose information regarding the user's location, nationality or language. Next, we compile a comprehensive list of mappings between countries and ethnicities, from online sources, and use it to automatically cross-match and verify that the inferred information indeed refers to locations and nationalities. By matching the detected entities with the ethnicities and countries in our list, we were able to automatically verify 1,945 extensions. Since our list does not contain region/city names, we manually inspected the remaining entities and found 315 additional extensions with descriptions that include information that could reveal the user's location or nationality. However, in our analysis we do not consider any entities that can reveal information but require region-specific knowledge by the attacker (e.g., UUID: `cgdogkoldofookodmiipcakaejpgocgc`). In total, this approach led to the identification of 2,260 extensions.

Next, by using our name-lists we map the names of the reviewers of each extension to their gender, and calculate the percentage between male and female. We find that for 1,448 extensions the percentage of one gender over the other exceeds 80%, which in many cases is sufficient to determine the gender of the users that have the extension installed in their browser.

Since Google's API cannot classify all the detectable extensions, as some of them have a very short description text, we opt for another approach that could identify extensions that reveal sensitive information. Thus, we use publicly available wordlists [17] of religious and medical terms, and search for those terms in the extensions' description text. For this task we first discard certain terms in the wordlists' terms that are generic or have multiple meanings (i.e., the terms *virus* and *infection* have a different meaning in the context of the Web), as they could lead to many false positives. This straightforward approach of matching terms returned 73 extensions that are related to religion and 70 that are health related. We manually inspected these extensions and found that indeed 58 (79.45%) of the former ones reveal the user's religion. For the latter we found that 62 extensions are related to health (88.57%) and that 49 of these (70%) reveal health conditions. The remaining 13 extensions are for physicians or web developers (e.g., to help them build websites that are suitable for colorblind users).

We also created a wordlist with political terms and used it to identify extensions that could possibly reveal the political inclination of the user. Intentionally, we keep this wordlist short, only containing terms that clearly refer to politics (such as Democrats, Republicans, Liberals, Conservatives, Donald Trump, Hillary Clinton, Obama, UKIP, Brexit, etc.). With this wordlist we matched 340 extensions, and though manual inspection we found that indeed 323 (95%) are related to politics and that 307 of them (90.29%) provide insights about the user's political inclination.

Overall statistics. To have a more complete view regarding the extensions that reveal sensitive information about politics, health and religion, we combine the results of the classification with the results of the wordlist-based approach (only the extensions that we manually verified as TPs from the wordlist-based approach) This results in 387, 147 and 116 extensions that reveal information about the user's political inclination, health and religion, respectively. Furthermore, we find that these extensions have been installed 406,869, 1,177,573, and 885,923, times respectively, highlighting the extent of this significant privacy threat. If we consider all the categories from Figure 5, since even less sensitive categories are useful for privacy-invasive practices like targeted advertising, the total number of installations exceeds 59 million.

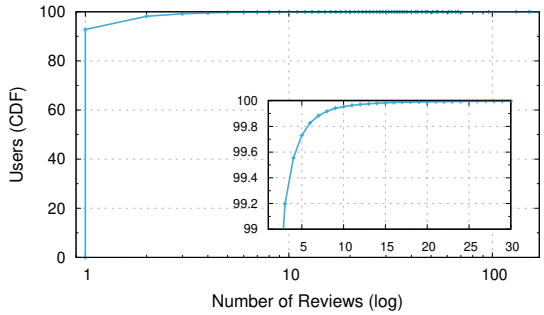

Fig. 6: Number of reviews per user.

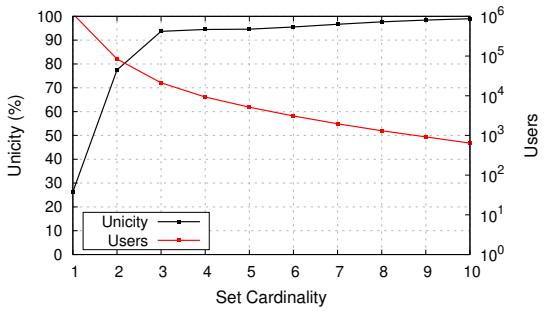

Fig. 7: Unicity of fingerprintable extension sets of different size, and the corresponding size of the anonymity crowd.

Furthermore, when considering all the above approaches for the inference of sensitive user information, from the name-based ethnicity and gender inference to the identification of the user's language, religion, political inclination etc., we find that 18,286 (62.13%) of the detectable extensions reveal such pieces of sensitive information. 15,996 of all these extensions can be identified through WAR and 14,042 of them cannot be identified through other techniques. Behavior-based fingerprints can identify 3,879 such extensions, of which 1,916 are not detectable otherwise. Lastly, 617 and 240 extensions can be identified through inter- and intra-communication fingerprints, and 134 and 52 cannot be detected by other means.

De-anonymization attack. Next we focus on the uniqueness of fingerprintable extensions for quantifying their suitability for identifying users solely based on their set of extensions. While prior studies explored how users could be uniquely tracked within an anonymous crowd, we demonstrate a more powerful attack that can infer the reviewer's name based on the uniqueness of their set of extensions. For our analysis we only use eponymous reviews, which also include a unique user ID – this removes the obstacle of users with identical names (the number of reviews per user in our dataset is presented in Figure 6). While we *do not* attempt to actually de-anonymize any users, in practice attackers could use the name and profile picture to discover even more information about the user [37]. Users can also be trivially matched to their reviews in other Google services (e.g., business, restaurants, etc.) which can lead to the inference of additional personal data. We want to emphasize that our de-anonymization attack is, obviously, only applicable to users that have written reviews for extensions; for other users the attacker would be limited to anonymous tracking as in prior studies.

We implement the unicity formula proposed by Achara et al. for calculating the uniqueness of smartphone apps [9], and use it to calculate the probability that a randomly selected subset of extensions with cardinality K is unique within our dataset of users. Prior work [52] reported that users had an average of 4.81 extensions, so we calculate the unicity for cardinality values of $K = 1, \ldots 10$, as shown in Figure 7. Surprisingly, we find that even when an attacker is able to only detect 2 extensions in a user's browser, there is a 77.5% chance of uniquely identifying the user within a set of almost 84 thousand users. As one would expect, as the cardinality increases so does the probability of uniquely identifying the user. When assuming that 4 extensions have been detected, 9,286 users are candidate targets with a 94.5% probability of being uniquely identifiable. When comparing to the numbers reported in [24] we find that our unicity results are higher for $K < 4$, which can be attributed to the significantly larger number of users in our study. In practice, the number of extensions installed per user will likely be higher than their number of reviews (i.e., users are unlikely to write a review for all their extensions) which could further increase their unicity. Nonetheless, due to our larger number of users and larger set of detectable extensions, we believe that our study offers a more accurate representation of the discriminatory power of browser extensions. Our findings also highlight the significant privacy risk that any type of public data can introduce, even something as innocuous as extension reviews.

VII. Discussion and Future Work

Ethical considerations. The techniques presented in this paper present a severe privacy risk to users. However, it is important to note that we do not actually run our attacks against any users. Our attacks are based on the analysis of extension characteristics and our goal is to explore what sensitive information can be inferred from the presence of such extensions. During our experiments we did not attempt to correlate any extracted traits/characteristics to users. Furthermore, the review analysis process relies on aggregate statistics regarding names collected from *publicly available reviews* from Chrome's Web Store. The unicity measurements leverage reviewers' unique IDs for associating users with installed extensions, and does not take into account or get correlated to any actual user information. We have deleted all collected reviews after running our experiments. We believe that our findings provide significant additional incentives for browser vendors to adopt defenses that have been recently proposed by the research community. Apart from extension enumeration techniques enhancing the uniqueness of a browser's fingerprint, our inference techniques mandate a reassessment of the extension ecosystem and the threat it poses to users, and motivate a more cautious approach to installing extensions.

Countermeasures. As demonstrated by our experimental evaluation, the countermeasure proposed by Trickel et al. [55] is ineffective against the vast majority of our behavioral fingerprints. Two other studies [46], [50] recently proposed whitelist-based countermeasures for mediating access between extensions and web pages, and a similar approach has been announced by Chrome [6]. These mechanisms can potentially reduce the fingerprint surface exposed to certain domains,

but the ones that users whitelist will still be able to run the attacks we demonstrated. While giving more control to users is a positive development, site-blocking mechanisms that rely on user configurations for setting policies can lead to user confusion [35] and may be too challenging for average users. Nonetheless, while more research is needed to fully prevent extension fingerprinting attacks, we believe that these approaches are important steps towards better protecting users and should be incorporated by browsers. While designing an effective countermeasure is out of the scope of this work, we believe that a technique that introduces innocuous (or imperceptible to the user) behavioral activity that results in the behavioral fingerprints of extensions resembling the fingerprints of other extensions is an interesting future direction.

Classification and information extraction. We mainly rely on the description text of the extensions for identifying each extension's topic and the sensitive information that can be possibly inferred. However, since there are no guidelines mandating the content and structure of the descriptions, these are determined solely by the developer of the extensions and are typically very inconsistent. Even though we developed a pre-processing method, we cannot remove all text that can possibly affect our classification results. As such, we plan to investigate more advanced techniques for identifying the relevant content, extracting the topic, and inferring sensitive user information (e.g., machine learning classifiers that take all extensions into account and detect more complex patterns).

Supplementary identity sources. We demonstrated how attackers could leverage extension reviews as a potential source for inferring a user's identity. In practice, users leave behind an abundance of digital "breadcrumbs" that result in privacy loss. These can be correlated [22] to further amplify the attack's effect. An adversary could also potentially augment the dataset of user reviews with reviews from other domains, namely mobile apps. By automatically mapping specific browser extensions to the corresponding mobile apps (e.g., the Skype Chrome extension to the Skype Android app) an attacker can use the additional reviews from other platforms to create more complete user profiles. Apart from using the name and images as identifiers, stylometric techniques [11], [43] can be used for correlating users across platforms.

Review analysis. During our inference attack analysis all available reviewers are considered as part of the user set for each extension regardless of the score that they have assigned. This provides a lower bound estimation of unicity as it could inflate the size of the user sets, which reduces the "uniqueness" of that extension. A more conservative approach is to use a heuristic based on the review score for assigning a user to the set of users that have installed that extension. However, users that have given a low rating may still continue to use that extension. As such, we plan to explore more sophisticated NLP-based techniques for identifying cases where users implicitly reveal that they have uninstalled a given extension.

VIII. Related Work

Browser fingerprinting [18] has garnered significant attention from the research community, and prior work has demonstrated the feasibility of several techniques that focus on different browser aspects of the browser and underlying hardware [13], [33], [40], [41]. More recently several studies have focused on the fingerprintability of browser extensions, which were also the focus of several blog posts by security researchers in the past [14], [23], [29], [30]. Sjosten et al. [47] presented the first large-scale study and demonstrated how extensions expose WARs which allow websites to identify and enumerate the extensions installed in a user's browser. At the same time Starov and Nikiforakis [52] proposed the use of DOM modifications as behavior-based fingerprints for extensions. They also conducted a user study with 854 participants and found that 14.1% of them had distinct sets of extensions that could be detected by any website, thus, uniquely identifying them. More recently, Gulyas et al. [24] conducted a large user study with more than 16 thousand participants and, using the fingerprinting technique from [47], found that 54.86% of users were uniquely identifiable based on their installed extensions. An alternative approach that relied on a timing-based side-channel attack was proposed by Sanchez-Rola et al. [44]. The core of their attack relies on the access control mechanism enforced by browsers to prevent unauthorized access of extension resources that have not been explicitly labeled as public, which implicitly reveals the existence (or absence) of a specific extension. A variation of their time-based attack was presented by Van Goethem and Joosen [56] as part of their exploration of fingerprinting attacks that can link users' isolated browsing sessions.

Apart from the fingerprintability of extensions, prior work also explored one dimension of privacy leakage due to extensions. Motivated by the seminal work of Krishnamurthy and Wills on privacy diffusion on the web [31] and leakage in request towards third parties [32], Starov and Nikiforakis built a dynamic analysis framework that detected the leakage of information (e.g., browsing history and search queries) from Chrome extensions to third parties [51]. Recent studies also demonstrated in different contexts how publicly available data could enable the inference of sensitive data [17] or lead to the de-anonymization of users [54].

IX. Conclusions

With browser vendors incorporating countermeasures against cookie-based tracking, user tracking techniques that rely on browser fingerprinting are becoming increasingly prevalent. As a result, modern browsers have recently introduced (or announced) mechanisms for mitigating the effect of such techniques. Nonetheless, recent research has exposed vulnerabilities in those countermeasures and have also proposed additional countermeasures. In this paper we presented the largest study on the unicity of extension fingerprints to date and revealed their discriminatory effect in real-world settings – apart from enabling attackers to uniquely identify a device and track users, we outlined a de-anonymization attack that leverages publicly available extension reviews for revealing the user's identity. We also conducted the first study detailing how attackers can infer sensitive or personal user information from detected extensions. The practicality of our attacks is highlighted by our comprehensive exploration of multiple extension fingerprinting techniques (including two novel approaches) and their evaluation under practical settings. Our experimental evaluation also demonstrated the robustness of our fingerprinting techniques against state-of-the-art countermeasures proposed by the research community, thus motivating the need for

additional research for potential countermeasures. Overall, we hope that our research sheds more light on the risks users face and leads users to a more critical view of extensions.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. Special thanks to our shepherd Adam Doupé for all his help. This work was supported by the DARPA ASED Program and AFRL (FA8650-18-C-7880), and NSF (CNS-1934597). Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the US Government.

REFERENCES

[1] "Chrome Developer Guide - Content Scripts," https://developer.chrome.com/extensions/content_scripts, accessed on 2019-12-30.

[2] "Chrome Developer Guide - Manifest - Web Accessible Resources," https://developer.chrome.com/extensions/manifest/web_accessible_resources, accessed on 2019-12-30.

[3] "Chrome Developer Guide - Message Passing," https://developer.chrome.com/extensions/messaging, accessed on 2019-12-30.

[4] "Google Cloud - AI & Machine Learning Products - Natural Language," https://cloud.google.com/natural-language/, accessed on 2019-12-30.

[5] "List of sensitive extensions," https://pastebin.com/ux0QKf5S.

[6] "Google security blog - trustworthy chrome extensions, by default," https://security.googleblog.com/2018/10/trustworthy-chrome-extensions-by-default.html, 2018, accessed on 2019-12-30.

[7] "Reuters - apple says uighurs targeted in iphone attack but disputes google findings," https://www.reuters.com/article/us-apple-cyber/apple-says-uighurs-targeted-in-iphone-attack-but-disputes-google-findings-idUSKCN1VR29K, 2019.

[8] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective: Dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13, 2013.

[9] J. P. Achara, G. Acs, and C. Castelluccia, "On the unicity of smartphone applications," in *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*. ACM, 2015, pp. 27–36.

[10] S. Bandhakavi, S. T. King, P. Madhusudan, and M. Winslett, "Vex: Vetting browser extensions for security vulnerabilities." in *USENIX Security Symposium*, vol. 10, 2010, pp. 339–354.

[11] M. L. Brocardo, I. Traore, S. Saad, and I. Woungang, "Authorship verification for short messages using stylometry," in *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE, 2013, pp. 1–6.

[12] A. S. Buyukkayhan, K. Onarlioglu, W. K. Robertson, and E. Kirda, "Crossfire: An analysis of firefox extension-reuse vulnerabilities." in *NDSS*, 2016.

[13] Y. Cao, S. Li, and E. Wijmans, "(cross-)browser fingerprinting via OS and hardware level features," in *24th Annual Network and Distributed System Security Symposium, NDSS*, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/cross-browser-fingerprinting-os-and-hardware-level-features/

[14] C. Cattani, "The evolution of chrome extensions detection," http://blog.beefproject.com/2013/04/the-evolution-of-chrome-extensions.html, 2013, accessed on 2019-12-30.

[15] Q. Chen and A. Kapravelos, "Mystique: Uncovering information leakage from browser extensions," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 1687–1700. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243823

[16] A. Datta, M. C. Tschantz, and A. Datta, "Automated experiments on ad privacy settings," *Proceedings on privacy enhancing technologies*, vol. 2015, no. 1, pp. 92–112, 2015.

[17] K. Drakonakis, P. Ilia, S. Ioannidis, and J. Polakis, "Please forget where i was last summer: The privacy risks of public location (meta)data," in *26th Annual Network and Distributed System Security Symposium*. The Internet Society, 2019.

[18] P. Eckersley, "How unique is your web browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, ser. PETS'10, 2010.

[19] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1388–1401.

[20] Explosion AI, "spacy: Industrial-strength nlp," https://spacy.io/, 2019.

[21] K. Garimella, O. Kostakis, and M. Mathioudakis, "Ad-blocking: A study on performance, privacy and counter-measures," in *Proceedings of the 2017 ACM on Web Science Conference*, ser. WebSci '17, New York, NY, USA, 2017, pp. 259–262.

[22] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13, 2013, pp. 447–458.

[23] J. Grossman, "I know what you've got (firefox extensions)," http://blog.jeremiahgrossman.com/2006/08/i-know-what-youve-got-firefox.html, 2006, accessed on 2019-12-30.

[24] G. G. Gulyas, D. F. Some, N. Bielova, and C. Castelluccia, "To extend or not to extend: on the uniqueness of browser extensions and web logins," in *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. ACM, 2018, pp. 14–27.

[25] M. A. Hearst, "Texttiling: Segmenting text into multi-paragraph subtopic passages," *Comput. Linguist.*, vol. 23, no. 1, pp. 33–64, Mar. 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=972684.972687

[26] U. Iqbal, Z. Shafiq, and Z. Qian, "The ad wars: retrospective measurement and analysis of anti-adblock filter lists," in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 171–183.

[27] C. Jackson and A. Barth, "ForceHTTPS: Protecting high-security web sites from network attacks," in *Proceedings of the 17th International World Wide Web Conference*, 2008.

[28] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, "Hulk: Eliciting Malicious Behavior in Browser Extensions," in *Proceedings of the USENIX Security Symposium*. USENIX, 2014.

[29] J. Kettle, "Sparse bruteforce addon detection," http://www.skeletonscribe.net/2011/07/sparse-bruteforce-addon-scanner.html, July 2011, accessed on 2019-12-30.

[30] K. Kotowitz, "Intro to chrome addons hacking: fingerprinting," http://blog.kotowicz.net/2012/02/intro-to-chrome-addons-hacking.html, 2012, accessed on 2019-12-30.

[31] B. Krishnamurthy and C. Wills, "Privacy diffusion on the web: a longitudinal perspective," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 541–550.

[32] B. Krishnamurthy and C. E. Wills, "Characterizing privacy in online social networks," in *Proceedings of the first workshop on Online social networks*. ACM, 2008, pp. 37–42.

[33] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 878–894.

[34] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu, "Xray: Enhancing the web's transparency with differential correlation," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 49–64.

[35] P. Leon, B. Ur, R. Shay, Y. Wang, R. Balebako, and L. Cranor, "Why johnny can't opt out: a usability evaluation of tools to limit online behavioral advertising," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 589–598.

[36] L. Liu, X. Zhang, G. Yan, S. Chen *et al.*, "Chrome extensions: Threat analysis and countermeasures." in *NDSS*, 2012.

[37] A. Malhotra, L. Totti, W. Meira Jr, P. Kumaraguru, and V. Almeida, "Studying user footprints in different online social networks," in *Proceedings of the 2012 International Conference on Advances in Social*

Networks Analysis and Mining (ASONAM 2012). IEEE Computer Society, 2012, pp. 1065–1070.

[38] A. Mathur, J. Vitak, A. Narayanan, and M. Chetty, "Characterizing the use of browser-based blocking extensions to prevent online tracking," in *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018)*, 2018, pp. 103–116.

[39] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, "Block me if you can: A large-scale study of tracker-blocking tools," in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, 2017, pp. 319–333.

[40] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of W2SP 2012*, May 2012.

[41] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, E. Weippl, and F. Wien, "Fast and reliable browser identification with javascript engine fingerprinting," in *Web 2.0 Workshop on Security and Privacy (W2SP)*, vol. 5, 2013.

[42] NLTK Project, "Natural language toolkit," https://www.nltk.org/, 2019.

[43] R. Overdorf and R. Greenstadt, "Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, pp. 155–171, 2016.

[44] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Extension Breakdown: Security Analysis of Browsers Extension Resources Control Policies," in *Proceedings of the 26rd USENIX Security Symposium (USENIX Security)*, August 2017.

[45] S. Sivakorn, A. D. Keromytis, and J. Polakis, "That's the way the cookie crumbles: Evaluating https enforcing mechanisms," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, ser. WPES '16. ACM, 2016, pp. 71–81.

[46] A. Sjösten, S. Van Acker, P. Picazo-Sanchez, and A. Sabelfeld, "Latex gloves: Protecting browser extensions from probing and revelation attacks," in *26th Annual Network and Distributed System Security Symposium*. The Internet Society, 2019.

[47] A. Sjösten, S. Van Acker, and A. Sabelfeld, "Discovering browser extensions via web accessible resources," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, ser. CODASPY '17. New York, NY, USA: ACM, 2017, pp. 329–336. [Online]. Available: http://doi.acm.org/10.1145/3029806.3029820

[48] P. Snyder, C. Taylor, and C. Kanich, "Most websites don't need to vibrate: A cost-benefit approach to improving browser security," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 179–194.

[49] I. F. Spellerberg and P. J. Fedor, "A tribute to claude shannon (1916–2001) and a plea for more rigorous use of species richness, species diversity and the 'shannon–wiener'index," *Global ecology and biogeography*, vol. 12, no. 3, pp. 177–179, 2003.

[50] O. Starov, P. Laperdrix, A. Kapravelos, and N. Nikiforakis, "Unnecessarily identifiable: Quantifying the fingerprintability of browser extensions due to bloat," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: ACM, 2019, pp. 3244–3250. [Online]. Available: http://doi.acm.org/10.1145/3308558.3313458

[51] O. Starov and N. Nikiforakis, "Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1481–1490.

[52] ——, "Xhound: Quantifying the fingerprintability of browser extensions," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 941–956.

[53] StatCounter, "Browser market share worldwide," http://gs.statcounter.com/browser-market-share, 2019.

[54] J. Su, A. Shukla, S. Goel, and A. Narayanan, "De-anonymizing web browsing data with social networks," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1261–1269.

[55] E. Trickel, O. Starov, A. Kapravelos, N. Nikiforakis, and A. Doupé, "Everyone is different: Client-side diversification for defending against extension fingerprinting," in *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, 2019.

[56] T. Van Goethem and W. Joosen, "One side-channel to bring them all and in the darkness bind them: Associating isolated browsing sessions," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[57] J. Wagner, "Assessing loading performance in real life with navigation and resource timing," https://developers.google.com/web/fundamentals/performance/navigation-and-resource-timing/, 2019, accessed on 2019-12-30.