Dirty Clicks: A Study of the Usability and Security Implications of Click-related Behaviors on the Web

Iskander Sanchez-Rola
University of Deusto
NortonLifeLock Research Group

Davide Balzarotti
EURECOM

Christopher Kruegel
UC Santa Barbara

Giovanni Vigna
UC Santa Barbara

Igor Santos
University of Deusto

ABSTRACT

Web pages have evolved into very complex dynamic applications, which are often very *opaque* and difficult for non-experts to understand. At the same time, security researchers push for more *transparent* web applications, which can help users in taking important security-related decisions about which information to disclose, which link to visit, and which online service to trust.

In this paper, we look at one of the simplest but also most representative aspect that captures the struggle between these opposite demands: a mouse click. In particular, we present the first comprehensive study of the possible security and privacy implications that clicks can have from a user perspective, analyzing the disconnect that exists between what is shown to users and what actually happens after. We started by identifying and classifying possible problems. We then implemented a crawler that performed nearly 2.5M clicks looking for signs of misbehavior. We analyzed all the interactions created as a result of those clicks, and discovered that the vast majority of domains are putting users at risk by either obscuring the real target of links or by not providing sufficient information for users to make an informed decision. We conclude the paper by proposing a set of countermeasures.

CCS CONCEPTS

• **Security and privacy → Browser security**.

KEYWORDS

browser click; web security; usability

ACM Reference Format:
Iskander Sanchez-Rola, Davide Balzarotti, Christopher Kruegel, Giovanni Vigna, and Igor Santos. 2020. Dirty Clicks: A Study of the Usability and Security Implications of Click-related Behaviors on the Web. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3366423.3380124

1 INTRODUCTION

Despite its current complexity, the World Wide Web is still, at its core, an interconnected network of hypertextual content. Over the years, static pages have been largely replaced by dynamic, stateful, web applications. However, links and other clickable elements still play a fundamental role in driving the interaction with users: it is by clicking on links that most users navigate from one website to another, and it is by clicking on menus, buttons, and other elements of the DOM that they interact with a page and trigger functions.

Unfortunately, browsing the web also introduces important security risks. In fact, it is through malicious and compromised web pages that many computers are infected with malware, and credentials and other personal information are regularly stolen from millions of users [23, 42]. On top of these criminal activities, online tracking, as performed by advertisement companies and other large corporations, is one of the main privacy concerns for our society [13, 50]. This translates into the fact that users need to be extremely careful when visiting webpages. For instance, it is very common to warn users not to *click* on suspicious links, and to always verify the different indicators provided by their browsers to alert about potentially dangerous targets. In 2015 Egelman and Peer [12] compiled a list of the most common computer security advises, and used this information to derive a Security Behavior Intentions Scale (SeBIS). One of the 16 final questions selected by the authors to assess the users' security behavior is "*When browsing websites, I frequently mouseover links to see where they go, before clicking them*". In particular, this factor is one of the only five selected to measure whether users are able to identify environmental security cues. Moreover, in a later user study by Zhang-Kennedy et al. [70] the authors found that more than half of their participants always/often check the links' URL before clicking on them. Even though this same security tip has been repeated countless times, no one to date measured to which extent this is possible – as bad web design practices can make this step impossible for users to perform.

In this paper, we look closely at this problem, and we measure how widespread are these bad practices, and whether they are becoming the norm rather than the exception. Most of the work performed to date on clicking behavior has focused on the server side, i.e., on how an application can identify if a click was actually made by a real user, and not by an automated machine or a script (the so-called "click fraud") [32, 41]. This is an important problem, especially in the context of the advertising pay-per-click (PPC) pricing model, but it is only a piece of a much larger picture. To fill this gap, our study looks at the click ecosystem from the user perspective, with a focus on the different security and privacy threats to which a user may be exposed.

We present an extensive analysis that sheds light on the most common click-related techniques used (intentionally or not) by web

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380124

developers. Despite the fact that one may expect bad practices to be more common in dubious web sites (such as those associated with free streaming [43] or porn [64]), our experiments show that their adoption is nearly identical in highly accessed webpages listed in Alexa [1]. Around 80% of the domains we tested adopt some form of misleading technique that would prevent users from making informed decisions on whether they want or not to click on a given link. Moreover, around 70% of the domains exposed users to unexpected man-in-the-middle threats, 20% of which were completely undetectable by a user even after the click was performed. Even worse, 10-to-20% of the time a link pointing to a low-risk website resulted in a visit to a site categorized as highly dangerous.

2 TOWARDS A CLICK "CONTRACT"

Today, there are no direct guidelines that completely define what is the acceptable behavior when a user clicks on an element of a web page. However, there are a number of important assumptions, which users and web developers often take for granted, that characterize such expected behavior. In order to formalize a *click contract*, we propose a number of rules that are based on previous web recommendations/standards and user experience handbooks.

Based on its definition [67], *"the href attribute in each source anchor specifies the address of the destination anchor with a URI"*. Therefore, websites should follow the World Wide Web Consortium (W3C) description, and use `href` to indicate the destination of the link. The `Same-document References` [65] then describes the case in which the URI reference is empty, and states that *"the target of that reference is defined to be within the same entity"*. Additionally, elements are identifiable as clickable [34] *"Used, e.g., when hovering over links. Typically an image of a hand"*, so if a retrieval actions is performed after clicking some element not marked as clickable [20, 66], they would be not using the defined method for it.

When designing and browsing websites, it is essential that they follow general user experience guidelines in order to make them usable and secure. In the specific case of clicks, we want to empathize the concept of *dependability* [54, 55], which indicates *"Does the user feel in control of the interaction? Can he or she predict the system's behavior?"*. More concretely, recent user-driven studies using this methodology [25, 30] define it as cases in which a link *"redirects the page to the right place and website and not redirecting to other websites"*. Based on this concept, secure channels could be ambiguous for users based on current indicators (e.g., green padlock) [36], they describe cases in which *"the connection has not been intercepted"*, and therefore should not be used when an intermediate website in a chain of redirections is unencrypted. We also extended this concept to consider user tracking and third-party trust, as users want to be aware of unexpected situations of this nature [31, 63], and even current regulations are pushing in that direction[14, 24].

We can summarize these points, which form what we call the *click contract*, around two main concepts: What You See Is What You Get (WYSIWYG), and Trust in the Endpoints. It is important to indicate, that according to our definition, we do not consider background third-party content/requests (e.g., AJAX communications) a bad practice, as it is the base for many client/server interactions, and does not play a role in deceiving the user. We formalize our click contract in the following:

What You See Is What You Get:

(1) When a user clicks on a link whose target URL is displayed by the browser at the bottom of the screen, she expects to navigate to that same destination. In case redirections happen afterwards as a consequence of the click, the user expects to remain within the same domain of the displayed URL, or the website she is on at the moment of clicking.

(2) If an object is clickable, but the browser does not show any domain at the bottom of the webpage, a user expects the click to generate some action *within* the current website and not to navigate to a different domain.

(3) The user does not expect any external navigation to take place when she clicks on a non-clickable element of the page (such as a simple text paragraph).

(4) When the user clicks an HTTPS link, she expects that the communication towards the target URL will be encrypted.

Trust in the Endpoints:

(5) If a user on a website A clicks on a link to a domain B, she does not expect any other domain, apart from A and B (or those included by them), to execute code in her browser.

(6) If cookies are created in the process that follows a click, the user only expects cookies from the domain she clicked, or from any of the third party domains included by it.

(7) If a new tab is opened by the browser after the user clicks on a link, the new tab should not be able to interact with the other tabs already open in the browser.

In the rest of the paper, we present a comprehensive measurement of how widespread are violations of these seven points in the Web. We will also identify and discuss potential security and privacy threats to which a user may be exposed due to the poor usability of websites that do not follow these practices.

3 REAL-WORLD EXAMPLES

In this section, we present two real examples of websites that suffer from some of the bad practices related to the *click contract*. These cases can help to better understand what website owners are doing, and what the potential consequences for the end users are. These examples were automatically discovered during our experiments, as we will describe in more details in Section 4.

The first case we want to discuss is the website of a prestigious university, that contains a page with a form to join the mailing list of one of its organizations. When a user clicks the submit button (which has no `href`), the page redirects to a different website owned by a company related to tracking services, and then this new website redirects back (thought JavaScript) to the original page. This final page is exactly the same as the one the user clicked, but with a "thank you" message on the top. The expected behavior in this case would have been that clicking on the submit button generated a POST request, and that a JavaScript listener was used to write the acknowledgment message. Instead, the user is redirected to an external company that executes JavaScript code without any control from the original website. We checked what this intermediate website did, and it created a long lasting identifier that would be accessible as a third-party cookie. Even if the user tried to avoid

unexpected identifiers choosing "Only Accept third-party cookies: From visited" in the browser [35], the identifier created in this case would still be accessible, as the user actually visited the website (even if she was unaware of the hidden redirection).

The second example is from a website offering betting discounts and tips. When the user clicks on a discount (with an `href` pointing to a subdomain of the website), she is redirected first to that URL, then to a subdomain of an external betting company, and finally to the promotional discount on the website of that same company. Both the second and third redirections are deceiving, as they result in the user visiting other third-party sites without her consent. But the main problem is in the second redirection. In fact, while the original `href` is HTTPS and the final website is also served over HTTPS, the intermediate subdomain access occurs over HTTP. Even worse, the intermediate connection is completely invisible for the user and therefore it is very difficult to detect this middle insecure transition. As the original website, the link clicked by the user, and the final destination use HTTPS connections, a visitor may erroneously believe that the entire chain was secure as well. Instead, the user is subject to a possible man-in-the-middle attack due to that intermediate HTTP connection. Moreover, she is also subject to a possible eavesdropper that can read all information sent on plain text. While analyzing this example for our case study, we realized that the user can even have her credit card indirectly compromised. In fact, the betting company does not create its login cookies with the `secure` attribute, and since the intermediate subdomain is HTTP, all those cookies are sent unencrypted. Therefore, a malicious actor could later reuse these cookies to access the account, which is linked to a credit card, and possibly use it to withdraw money from it.

4 DATA COLLECTION

To capture a view of the global *click ecosystem*, we gathered a dataset that includes top ranked domains (according to the Alexa domains list [1]) as well as domains belonging to more dubious categories, such as those offering the download or streaming of illegal content, or those serving adult and pornographic material. Our hypothesis is that popular websites would be less deceptive with their click-related behavior, while websites associated to one of those gray categories, can be more unpredictable and would tend to introduce more risks for the end users. For example, previous studies that analyzed the security of free streaming webpages [43] observed various situations where multiple overlays were used, superimposed on each other in the website, in order to generate unintentional clicks for certain links. Another recent study published in 2016 [64] found that many pornographic websites were redirecting users through JavaScript instead of using `href`, making it difficult to infer the final destination of the link by looking at its URL. Because of these preliminary findings, we conducted experiments to verify whether these poor practices are more prevalent in these classes of websites with respect to the rest of the Web.

4.1 Domains Selection

We started by populating our gray list performing a number of different queries, focusing on illegal content (either video streaming of software download) and pornographic pages, and using the autocomplete feature offered by search engines (e.g., *"game of thrones season 7 free download"*). In particular, we performed five different queries for each of the following eight categories: series, movies, music, games, software, TV, sport events, and adult content. To increase the coverage of our domain retrieval phase, we executed each query in four different search engines (Google, Bing, DuckDuckGO, and Yandex) and we stored the first 100 links returned.

Moreover, to avoid incurring into very popular websites, we filtered this preliminary list of collected domains by removing those that also belonged to the Alexa Top 1k category, and we performed a manual sanity-check to verify that the resulting domains indeed belonged to the categories depicted above. This resulted into a gray dataset containing 6,075 unique domains.

We then randomly selected the same number of domains from the Alexa's Top 10k, Top 100k and Top 1M lists (2,025 each). By combining both the Alexa domains and the gray domains, we obtained a final dataset of 12,150 unique domains for our experiments.

4.2 Analysis Tool

We implemented our click analysis tool using a custom crawler based on the well-known web browser Chrome. The crawler receives as input the main URL of a website, loads the corresponding page, and then recursively visits three randomly selected pages up to a distance of three clicks from the home URL. This results in the analysis of 13 pages per website, mimicking a configuration previously used by other researchers in similar studies [49].

It is important to remark that we consider to be "clickable" all elements that have the `cursor` property set to `pointer`. As defined by Mozilla [34]: *"The element can be interacted with by clicking on it"*. Some elements have it by default, such as anchor links with `href`, others need to have it explicitly indicated, or inherit it from their parent element. While it is possible for elements to react to a click even without setting a different cursor, this is per-se already a deceiving behavior. In fact, a user may decide to click on some text to select it, and she would not expect this to trigger her browser to navigate to another page. Therefore, we considered this phenomenon in Section 5, where we measure how many websites adopt this technique to capture unintended user clicks.

On each visited page, our crawler performed 21 different clicks. The first is executed over a randomly selected seemingly non-clickable element, with the goal of identifying websites that contain an invisible layer that intercept the user's clicks. To avoid the impact of such invisible layers in the rest of the tests, polluting the click analysis, we maintained the same session between every consecutive click on the same page.

The tool then dynamically computes the appearance of all clickable objects according to styles defined both in the CSS stylesheets and in the `style` tags embedded within the HTML. It then uses this information to rank each link according to its computed visualization size and performs one click on each of the ten largest elements. Finally, it concludes the analysis by randomly clicking on ten other clickable objects. In total, this process results in up to 273 clicks for each website (21 per page). In order to avoid mis-classifying websites according to their advertisements, or incurring in a possible click fraud, we instructed our crawler not to click on elements

directly linked to advertisement companies, as indicated by the list used by Mozilla Firefox [38].

The crawler captures and records on a separate log file the entire behavior both during and after a click is performed. This information is retrieved by using the Chrome debugging protocol, which allows developers to instrument the browser [8]. To evade the detection of our automated browsing, we implemented the most recent methods discussed in similar studies [11, 52, 53]. Our instrumentation is divided in multiple groups (e.g., DOM and Network) that support different commands and events. Following this procedure, our tool is able to perform mouse click events natively, and precisely detect all the possible situations it can create. For instance, we can detect when a new tab is created through the `targetCreated` event or retrieve created coookies using the `getCookies` function.

There is a clear trade-off between the accuracy of the results and the scalability of the measurement process. As a result, it is possible that some of the websites for which we did not discover any anomalous behavior were actually performing them, but only on a small subset of their links. We will discuss in more details the coverage of our measurement in Section 6 and the consequences for the precision of our results in Section 8.

4.3 General Stats

Our crawler performed a total of 2,331,239 distinct clicks in 117,826 pages belonging to 10,903 different web sites – 5,455 of which belonged to the Alexa top-ranked domains and 5,448 of which belonged to the gray domains, showing a balanced dataset between the two main categories. 1,247 web sites could not be analyzed because they were offline, replying with empty document, or without any clickable element. Since not every domain has 13 different pages with at least 21 clickable elements each, the final number of clicks is slightly smaller than the result obtained by multiplying the individual factors. Additionally, as some advertisements may not include a domain in the `href` in order to hide their nature, we used the corresponding accesses generated after the click to detect these cases. We removed a total of 42,663 clicks following this process. We believe our dataset is sufficient for this specific analysis, in particular given the widespread adoption of the threats.

It is interesting to observe that, on average, for each website our analysis covered 28.32% of all clickable elements. From all the clicked objects, 72.33% had an `href` attribute that displayed to the user a target URL location associated to the element. The remaining 27.07% did not indicate this information, suggesting that the target resided in the same domain of the currently accessed webpage. Interestingly, only 42.19% of the links with an `href` and 45.39% of those without used the secure transfer protocol (HTTPS).

5 FINDINGS

There are many security and privacy implications involved when a user clicks on an element in a webpage. In this paper, we focus on a particular aspect of those risks, namely the fact that the user has enough information to take an informed decision on whether or not she wants to proceed with her action. For instance, if a user clicks a link with a `href` attribute pointing to an HTTP webpage as destination, she consciously accepts the risk of receiving data in the clear over the network. However, things are different when

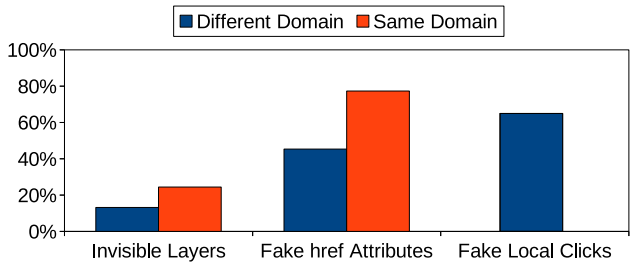

Figure 1: Percentage of domains misleading users.

Table 1: Occurrences of webpages misleading users.

Type	Total Occurrences	Targeting Different Domains
Invisible Layer	19,696	54.33%
Fake `href` attributes	138,860	31.14%
Fake local clicks	123,959	100.00%
TOTAL	282,515	63.00%

the same user clicks on a link with a `href` attribute pointing to an HTTPS URL but the web application decides instead to issue the request over the HTTP protocol. The final result remains the same (in term of communication over a cleartext channel), but in the second scenario the user had no information to take an informed decision, and was deceived into believing her data would be transmitted over a secure channel.

In this section, we present threats that the users could not predict before clicking, as they are much more dangerous and difficult to detect even for experienced users with a security background, due to the lack of information required to perform any preventive actions. All the results shown in this section are calculated from aggregated data from both datasets used in this work. After performing various statistical tests, we found that both datasets share the same properties regarding click implication occurrences. We will explain and discuss these statistical tests in Section 6.

While the issues discussed in this paper can lead to actual security risks, as we will discuss in more details in Section 7, it is important to remark that our goal is mainly to measure the disconnect that exists between the information that links present to the users and the actions associated to their clicks. This difference completely undermines one of the most common and repeated security advice: to look at the URL before clicking on a link [12, 61, 70].

5.1 Misleading Targets

One of the most important aspects for the user when performing any type of click in a webpage, is trust. Trust implies that when the webpage explicitly mentions the target URL, this is indeed where the browser will navigate to [66, 67]. Even though many users take this trust for granted, webpages do not always follow this rule and often mislead users into performing actions that are different from

the intended ones. In our study, we have detected three different types of misleading clicks:

- **Invisible Layer:** The user clicks some non-clickable object of the webpage (e.g., some random text or image), despite the fact that there should not be any expected result, this triggers a webpage redirection or the opening of a new tab.
- **Fake `href` Attributes:** The user wants to click on a given element, such as a simple `<a>` tag, and the user's expectation is that the browser will go to the website indicated by the link (as specified in the `href` attribute). However, the user is redirected to a different website, not related to the expected one.
- **Fake Local Clicks:** The user clicks on a clickable object in a webpage that does not explicitly indicate a target URL. As a result, the user expects the destination to be in the same domain of the current website [65]. However, the user is redirected to a completely unrelated domain without any prior notice.

Results. As shown in Figure 1, roughly 20% of the websites contained an invisible layer that captured the user's clicks. Moreover, more than 10% of all websites are redirecting the user to a completely different domain in this case. If we check the global numbers (Table 1), we can see that more than half of all the redirections/new tab opens using this technique were performed to a different domain. Our data shows that this is a very widespread problem and that in the majority of the cases the target URL is not even located on the same domain.

Figure 1 also shows that the vast majority of websites (nearly 80%) mislead users by reporting incorrect `href` attributes on some of their links. Even worse, in over 45% of the cases those links pointed to completely different domains from those reported in the displayed URL. Finally, fake local clicks are also quite common on the web with 65% of the websites we tested (Figure 1) adopting this technique. Interestingly, the total number of occurrences is the same as the fake `href` attributes, showing a similar global trend between both techniques (Table 1).

To sum up, misleading targets are worryingly popular among all types of websites. In fact, despite the common intuition that this type of techniques would be prevalently used in gray webpages for aggressive advertisement reasons, our results show that most of these bad practices are equally common in both datasets.

5.2 Users Redirection

Even when a click initially behaves as expected, it is still possible for the user to be redirected to different pages without her consent. Of course, redirections are very common on the Web and can be used for perfectly legitimate reasons. Moreover, if a web page `a.com` contains a link to `b.com`, which will eventually redirect to another domain, the owner of `a.com` has no control over this behavior. Nevertheless, we decided to measure and report how prevalent this behavior is because, from a user point of view (pointed out in user experience guidelines [54, 55]), it still results in hiding the final target of a click. Ignoring internal (i.e., to the same website) redirections, we can classify the remaining redirections in:

- **Different Domain:** This family includes all the redirections to domains different from the one that the user was expecting to visit when performing the click [25, 30]. For example, if the user clicks a link on `a.com` pointing to `b.com`, any redirection involving any of the two domains is considered legitimate. This is the case in which `b.com` uses a redirection to point to another URL in the same website. However, if the users clicks on a link to `b.com` and ends up visiting `c.com`, this can potentially be deceiving.
- **Hidden Domain:** This is a more severe variation of the scenario described above. In this case, the user clicks on a link pointing to `b`, which *temporarily* redirects to `c`, which then in turn immediately redirects back to `b` – thus introducing a third domain in the redirection chain that the user would not even be aware of (as the browser would likely not show this intermediate step).

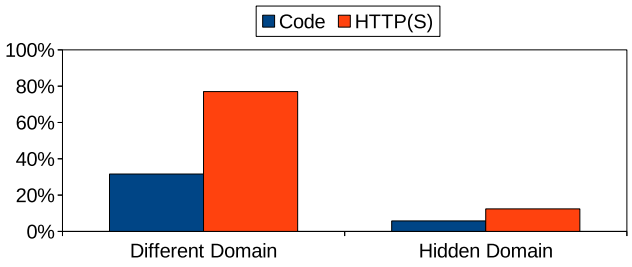

Figure 2: Percentage of domains redirecting users.

Table 2: Occurrences of webpages redirecting users.

Type	Total Occurrences	HTTP(S)	Code
Different Domain	525,975	68.68%	31.32%
Hidden Domain	42,558	31.31%	68.69%
TOTAL	568,533	65.88%	34.12%

On top of these two classes, there is another orthogonal classification related to the specific method used to perform the redirection. On the one hand, we have the HTTP(S) redirection, where the request can for example include the `Set-Cookie` header to create different cookies in the user's browser for that specific domain. The HTTP code employed in these redirection is `30X`, where the last number specifies the reason for the redirections (e.g., `302` is used to notify that the requested resource has been *Moved Temporarily*). On the other hand, we have code-based redirections that do not happen by means of an HTTP request, but by code being executed on the webpage, once it is parsed and loaded by the browser. The problem in this type of redirection is that the domains involved can execute JavaScript code without any control of the original or expected website (e.g., creating tracking identifiers). They rely on HTML refresh using a `meta` element with the `http-equiv` parameter, directly with JavaScript using `window.location`, or any other equivalent method. Even if header-based redirecting parties could change themselves to a code-based redirection, we checked how many are actually getting these privileged rights.

Independently from the method used to redirect the browser, for our study, we are particularly interested in how transparent it is to the user which domains have been visited during the transition, in particular in the case of multiple consecutive redirections.

Results. As shown in Figure 2, 80% of all domains perform HTTP(S) redirections pointing to completely different domains with respect to the ones expected by the users. Regarding code redirections to different domains, an impressive 35% of them use this technique. This is particularly worrying because of the aforementioned security problems, which may result in possible uncontrolled code executions or cookies. The user was never notified that she was going to give these rights to those domains. According to the global occurrence data presented in Table 2, the percentages follow a similar trend, with a majority of domains redirecting through HTTP(S) and a not negligible one third of domains allowing code execution.

More worryingly, around 15% of the analyzed domains stealthily allows other domains to gain uncontrolled cookie or code execution rights, by including them in the middle of redirections chains that end in the correct domain. Nearly 10% of them actually allow intermediate hidden domains to execute code without any control. Checking the total occurrence numbers (see Table 2), this percentage is much bigger, with nearly 70% of the websites allowing hidden domains to execute their own code. The problem here is very serious, as all hidden domains (not detectable for the user) that are using code redirections can execute JavaScript without any control from the original or expected website, allowing them to execute anything they want in the user's browser (e.g., tracking and profiling the user) The user was never informed that she was going to give these rights to those domains.

5.3 Insecure Communication

Man-in-the-middle attacks that can violate the user's privacy, steal credentials, and even inject/modify the data in transit are a serious threat to web users [6, 68]. When a user visits a website over HTTP, she implicitly accepts the fact that her traffic would not be protected against eavesdropping. However, when a user clicks on a link that displays an HTTPS URL, she expects to send her data over a protected channel [36, 54, 55]. Unfortunately, in reality we found that this behavior is not the rule. In particular, we identified three main scenarios in which this requirement is not met:

- **Insecure Access:** This is the basic case in which the user clicks an element pointing to an HTTPS URL but eventually the browser (either from the beginning, or because of a redirection) drops the secure channel and ends up visiting a page over an insecure HTTP connection.
- **Hidden HTTP Connection:** In this very subtle scenario, the user initially clicks on an HTTPS URL, and eventually lands on a website served over HTTPS. Everything may therefore seems normal, but unfortunately there were intermediate HTTP webpages (invisible to the user) visited by the browser before reaching the final destination. In other words, the two endpoints are secure but the entire communication was not – without the user being aware of it.
- **Unexpected Mixed Content:** By default, over a secure connection, browsers block what is generally known as active mixed content, i.e., elements served over HTTP that can directly interact with the content of the page. However, other elements such as images and video files (i.e., passive mixed content) are allowed [10, 37]. This opens the door to possible security and privacy attacks that use passive mixed content. For instance, an element loaded via HTTP can be modified to a 401 *Unauthorized* response that includes a `WWW-Authenticate` header asking for a confirmation of their credentials (which will be sent directly to the attacker) [46]. It is important to stress the fact that we are not analyzing the problems of mixed content in general [7], but the occurrence of this threat related to clicks. Following our usual guidelines, we only measure mixed content loaded in webpages from domains that are different from those that the user was aware of contacting.

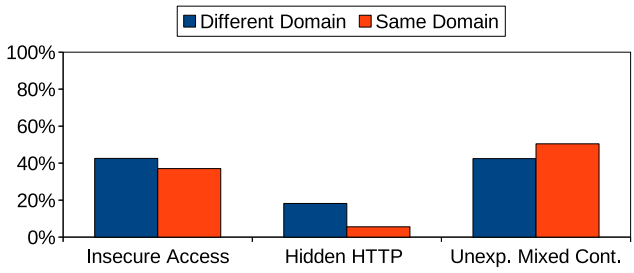

Figure 3: Percentage of domains creating man-in-the-middle threats.

Table 3: Occurrences of webpages creating MitM threats.

Type	Total Occurrences	Unique Domains
Insecure Access	185,984	23,570
* Different Domain	129,710	9,256
Hidden HTTP Connection	43,773	7,292
* Different Domain	39,903	2,484
Unexpected Mixed Content	279,550	22,322
* Different Domain	194,019	17,093
TOTAL	465,534	45,892

Results: Figure 3 shows that approximately 40% of all the domains we tested contained at least one link in which they insecurely redirected users over an HTTP connection when they explicitly specified HTTPS in the destination URL. To make thing worse (see Figure 3), a non-negligible 20% of these insecure redirections happen in the middle of theoretically secure connections, making it impossible for the end-user to detect this dangerous behavior. Overall (see Table 3), 23,570 unique domains were involved (sum of unique domains per accessed domain), and 30.94% of them were related to intermediate undetectable insecure HTTP connections.

Regarding the non-informed mixed content fetched from third-party websites, we measured that around 45% of all domains have at least one in their redirection chains (see Figure 3). In fact, only 5% of the domains include mixed content only from the same domain

Table 4: Occurrences of webpages opening new tabs.

Type	Total Occurrences
Link (_blank)	239,628
JavaScript (window.open)	613,457
TOTAL	853,085
* Protected	1,324

— the one that is expected and accepted by the user. This shows that more than half of the domains indirectly put their users in jeopardy not by performing an insecure redirections, but by loading external content over an insecure channel. Furthermore, if we count the unique domains that suffer from this problem, from a total of 22,322 different domain, a remarkable 76.57% belong to completely different domains of those expected by the user (as shown in Table 3).

5.4 Phishing-related Threats

While phishing attacks are usually associated with spam or scam campaigns, it is also possible for users to encounter a phishing website when surfing the Web. In this section, we explore how many websites are jeopardizing their visitors through their poor links hygiene. In fact, when a website opens a new browser tab or a new window, this new page obtains a reference to the original website that has triggered its opening through the `window.opener` object. To prevent the new site to tamper with the content of its parent, modern browsers are equipped with blocking capabilities through specific cross-origin actions derived from the well-known same-origin policy. However, it is still possible for the new tab to redirect the original opener website using the `window.opener.location` object, thus bypassing this protection [39].

In this way, from a newly opened tab, a miscreant is capable of detecting the domain of the opening website (by checking the HTTP `referer` header), and then redirecting the user to a phishing website of that same domain (maybe adopting some typosquatting techniques [33, 60] to make it harder for the user to notice the replacement), and finally even closing the new tab. For example, a user on Facebook can click a link to an external website that could act perfectly benign except from replacing the Facebook page itself with a fake copy that may be used to phish users into disclosing personal information or login credentials. This makes the scheme very difficult to detect even for an expert user. This type of attack is popularly called as "tabnabbing" [40, 45].

A simple solution exists to protect against this type of attacks: when a website includes links to external resources, it can specify `rel="noopener noreferrer"` to prevent the new page from accessing the parent URL [5, 19]. Equivalently, when a new tab is opened via JavaScript, by opening an `about:blank` tab, setting the new window's opener to `null`, and then redirecting it would solve the problem. However, still today many webpages do not adopt any protection methods when opening new tabs, exposing themselves and their visitors to these phishing attacks.

Results. During our experiments, a stunning 90% of the websites contained links that opened new tabs as a result of a click. Overall, this accounted for 853,085 new tabs. As reported in Table 4, the majority of them (71.91%) were opened by using JavaScript code.

Although this behavior is extremely widespread, we found that only 2% of the examined domains employed prevention techniques to secure their users from potential phishing attacks. For all links (see Table 4), the number is even smaller with only 1,324 protected links out of more than 850K visited ones.

In summary, these results show that nearly all of the new tabs opened are completely unprotected from possible phishing attacks. Moreover, opening new tabs is an very common action that most webpages do at some point.

5.5 User Tracking

One of the biggest concern nowadays regarding web privacy is web tracking, which consists in the ability to obtain or infer the users' browsing history, or to identify the same user across multiple different accesses. The first and still most common method to perform web tracking is based on cookies. In its most basic form, when a user visits a website `a.com`, she acknowledges that several cookies can be created and stored in her computer. These cookies can be set from the website she is visiting (`a.com`) or from a third-party domain (e.g., `z.com`) that may be also present on other websites (e.g., `b.com`, and `c.com`). This allows `z.com` to follow the user activity if she also visits these webpages. While Libert recently found [27] that in most cases the main domain does not notify the user about those third-party cookies, in this paper we take an optimistic position and we consider those cases as benign. What we are instead interested in measuring is the fact that the user is not even aware of new cookies generated [31, 36, 63], in the following cases:

- **Undesired Cookies:** If a user clicks on a link to `a.com`, she does not expect any other cookie besides the ones created by `a.com` and its direct third parties. Thereby, we will consider as undesired any cookie that does not follow this simple rule. For example, imagine that the previous click redirects you to `b.com` and later, though JavaScript, to `c.com`. All cookies set by `b.com`, `c.com`, and their respective third parties would be considered as undesired cookies.
- **Undesired HTTP Cookies:** In several cases, the problem is bigger than just having a large number of undesired cookies created in the browser. Sometimes, these cookies besides being undesired, they are also insecure, even if the user clicked a link directing to a secure webpage. For instance, a miscreant can perform a man-in-the-middle attack, and steal those cookies or even modify them to allow for future attacks or perform tracking of this user.
- **First-Party Bypass:** Browsers started introducing a new option to control the type of cookies they accept [2, 35]: accept cookies from the domain the user is currently visiting, but only allows third-party cookies from webpages previously visited by the user. Nevertheless, the current click ecosystem may undermines this option, as the user ends up *unintentionally* visiting many domains – which will therefore be whitelisted,

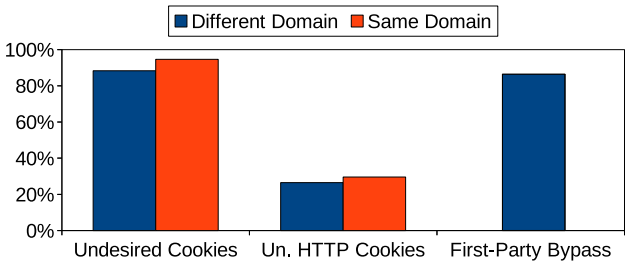

Figure 4: Percentage of domains creating tracking threats.

Table 5: Occurrences of webpages creating tracking threats.

Type	Total Occurrences	Unique Domains
Undesired Cookies	1,924,371	188,992
* Different Domain	1,241,806	165,735
Undesired HTTP Cookies	80,494	19,338
* Different Domain	73,171	18,175
First-Party Bypass	500,073	104,075
TOTAL	2,504,938	312,405

and allowed to set cookies. WebKit implemented a specific detection for these cases [62], but others browsers do not make any direct mention to this unwanted situation.

Results. In our experiments we did not count the number of cookies, but the number of domains that created undesired cookies. For example, if `b.com` created 5 undesired cookies and `c.com` 3 undesired cookies, we would report 2 (`b.com` and `c.com`) in our statistics (see Table 5). Moreover, unique domains are counted as the sum of unique domains per accessed domains.

The overwhelming number of domains (around 95%) created undesired cookies (see Figure 4). Globally, 64.53% of all occurrences were created by different domains, making a total of 188,992 unique domains. Analyzing the specific case of insecure undesired HTTP cookies, the number are much lower, but still concerning, due to the security and privacy problems they incur. 30% of domains created these type on dangerous undesired cookies, and our data shows that 90.90% of all the occurrences were performed by different domains (18,175 unique ones). Finally, we found 500,073 occurrences of unexpected domains becoming first-party webpages (104,075 unique), and thereby bypassing the newest cookie control policy implemented in browsers. Figure 4 shows that 87% of the websites (both in the Alexa and Gray categories), once visited by a user, as a side effect result in at least one new domain being added to the whitelist. As these domains were not visible to the user at any point in time before the click (and often even after), the user is completely unaware that they are considered "visited webpages" from now on.

6 STATISTICAL ANALYSIS

In Section 5, we analyzed (i) the percentage of websites that suffer from each problem we discussed in this paper, and (ii) the number and type of these occurrences. We now present the results of a number of statistical tests that show that both the Alexa and the gray domains categories follow similar trends in these practices.

For this specific case, conducting a *Chi-Square* test is the most appropriate approach, as the variables under study are categorical, and we want to check if the outcome frequencies follow a specific distribution. Following this method, we tested the null hypothesis that that the variables are independent. This way, we can compute the probability that the observed differences between the two groups are due to chance (statistical significance). If the corresponding p-value is larger than the alpha level 0.05, any observed difference is assumed to be explained by sampling variability. We found that many of the threats we presented have some statistical differences between the two groups. Nevertheless, with a very large sample size, a statistical test will often return a significant difference. Since reporting only these values is insufficient to fully understand the obtained results, we additionally calculated the effect size (*Cramer's V*) to check whether the difference is large enough to be relevant. In statistics, the effect size is a quantitative measure of the magnitude of a phenomenon, used to indicate the standardized difference between two means (the value should be greater than 0.15 in order to obtain an appreciable difference). Even if the difference is statistically significant in some cases, the effect size is virtually zero in all of them. This indicates that the actual differences are not large or consistent enough to be considered important, which confirms our statement that both groups follow similar trends.

7 THREAT RISKS

In a recent user study about security beliefs and protective behaviors by Wash and Rader [61], one of the questions was *"Being careful with what you click on while browsing the Internet makes it much more difficult to catch a virus."* In this section we check whether this this is the case by investigating the actual risks associated to the threats we measured.

In order to obtain this information, we used the risk level calculator for secure web gateways offered by Symantec [58, 59]. The service uses cloud-based artificial intelligence engines to categorize websites by using different indicators, such as historical information, characteristics of the websites, or features extracted from the server's behavior. Websites are classified in five risk groups, namely:

- **Low:** Consistently well-behaved.
- **Moderately Low:** Established history of normal behavior.
- **Moderate:** Not established history of normal behavior but neither evidence of suspicious behavior.
- **Moderately High:** Suspicious behavior (including spam, scam, etc.) or possibly malicious.
- **High:** Solid evidence of maliciousness.

It is important to remark that we did not analyze the websites in our dataset, but the websites the user was expecting to visit and the ones she accessed unintentionally because of the click threats presented in this paper. We then compared the risk level of the website that the user was expecting (e.g., `b.com`, low risk) with the website the user actually ended up accessing (e.g., `c.com`, high risk). Based on this, we computed two different factors, one indicating an increase in the threat risk, and another indicating an increase from

Table 6: A comparison between Alexa and gray websites according the increase in risk generated by the user click. The p-value is always lower than 0.05, indicating statistical significance for all values in this table.

	Alexa Websites		Gray Websites		Effect Size	
Click Implication Type	**Increase**	**Low to High**	**Increase**	**Low to High**	**Increase**	**Low to High**
Invisible Layer	43.07%	16.84%	58.49%	25.36%	**0.440**	**0.429**
Fake `href` Attributes	41.17%	5.42%	55.63%	18.92%	**0.229**	**0.268**
Fake Local Clicks	22.44%	4.55%	26.24%	8.41%	0.062	0.098
Redirecting	42.73%	9.85%	53.10%	23.42%	0.145	**0.222**
Hidden Domain	9.73%	0.63%	12.56%	3.05%	0.106	0.137
Insecure Access	66.01%	9.06%	74.74%	20.84%	0.141	**0.203**
Hidden HTTP Connection	35.79%	4.65%	47.90%	17.48%	**0.188**	**0.258**
Unexp. Mixed Content	41.99%	6.07%	39.94%	11.19%	0.051	0.117
Undesired Cookies	64.92%	15.18%	67.40%	29.86%	0.042	**0.232**
Undesired HTTP cookies	68.08%	12.89%	70.32%	25.95%	0.041	**0.216**
First-Party Bypass	50.10%	11.01%	59.84%	25.13%	0.141	**0.231**

the 'green' part of the spectrum, to the 'red' part. The percentages shown in Table 6 are the percentage of websites in each category that suffered from at least one case of the implications.

Overall, the consequences of the results of this test are very serious. For instance, fake `href` or redirections associated to a low-to-high risk transitions (which capture the cases in which a user clicks on a link considered safe by security products but ends up instead on a website flagged as malicious) account for 5-10% of the cases in the Alexa category and up to 19-23% in the gray group. In total, we detected that around half of the websites that have poor click hygiene actually increased the risk of the users because of these poor practices, and in 8.74% (for the Alexa set) and 19.33% (for the gray set) of the cases, the risk associated with the affected URLs went from "low" to "high".

Moreover, we statistically checked if the differences found between Alexa and gray websites for this factors were significant. We followed the same procedure as in the previous case, using *Chi-Square* and *Cramer's V* (see Section 6 for more details). In this case, all test showed a statistical significance. Moreover, the effect size scores are also considerably larger (often surpassing 0.15), showing that there is a clear difference between the two groups. These figures also show another important message. In fact, while we discovered that popular websites are no less deceptive than websites serving porn or illegal content, when these poor practices are present in the second group they are more often associated to a drastic increase in the risk for the users.

8 DISCUSSION

There are two main possible explanations for each of the different threats presented in this paper: (i) the flaws were deliberately introduced by the developers, or (ii) they were just the unintended consequence of poor practices or coding mistakes. While it may be difficult to know for sure, we believe that most cases fall into the second category. To clarify this statement, we are going to analyze the case studies presented in Section 3 from this perspective.

The case in which a form on the website of a prestigious university redirects to a external website without prior notice is the perfect example. It looks like the web developers wanted to collect some statistics of who was joining the mailing list, but instead of including the code themselves, they decided to rely on an external tracking company. This company might have asked the developers to include few lines of code in their website, probably without explaining the possible consequences of that action. As a result, there was probably no malicious intent, and the entire example is probably the result of a mistake by the site developers.

In our second example, the website used an intermediate subdomain in order track who was clicking on the offered discounts, probably without realizing that by doing that, the user could not tell anymore the final destination of her clicks. This is already per se a poor practice, but the problem goes one step further due to the hidden HTTP redirection. This is bad for two reasons. On the one hand, the website where the user is clicking should have checked if the redirection could be secured or not. On the other hand, the final betting website should either set its core cookies with the `secure` attribute, or implement *HTTP Strict Transport Security* (HSTS) to avoid this undesired intermediate insecure communications.

While plausible, the previous explanations are completely fictitious. In fact, it is impossible to know if the web developers were aware of the threats created and proceeded anyway, or if they did not realize the consequences of their actions. Because of this, as we will explain in more details in Section 9, we believe it is important to provide a service that web developers can use to analyze their own websites to detect the presence of poor practices.

8.1 Precision

Following the click analysis structure presented in Section 4, we performed nearly 2.5M different clicks. If we calculate the percentage of clicks we made comparing to all the possible clicks in each domain and compute the mean, we obtain 28.32% – which means than in average we clicked one third of the clickable elements in the pages we visited. We also calculated the percentages of clicks per domain that were affected by the various problems we identified in Section 5 and computed the values corresponding to different quartiles (e.g., Q3 and Q4) to obtain a general overview.

With the data relative to the quartiles and the percentage of total clicks performed, we can statistically estimate the probability of

detecting at least one case of every dangerous category with the amount of clicks we performed in a given website. In fact, we can model a website as an urn containing links of two categories: those affected by a given problem X and those that are not. Since we can estimate the percentage of the two types of links based on the data we collected, and we know that for a certain website we randomly visit (i.e., extract from the urn) a certain number of elements over the total number contained in the urn, we can estimate the odds of picking at least one link affected by the problem [4]. We repeated this computation for all the types of problems discussed in the paper. In average, the probability of misclassifying a website just because we did not test the right link varied from 0% (for tracking-related threats) to 4.7% in the case of insecure communications. These values show that when a website suffers from a poor behavior related to its links, this often affects a large percentage of its elements, thus making our sampling rate of testing one out of three links appropriate to estimate the presence of the different problems.

9 COUNTERMEASURES

In our measurement, we identified several bad practices on how click-related events are managed by existing websites. Even if some of them may have been deliberately introduced by the developers (e.g., to avoid recent cookie-control policies), we believe that the main cause for these problems is a lack of awareness, a lack of clear guidelines, and a poor understanding of the risks that these problems can introduce.

We hope that this paper can raise awareness about the widespread adoption of misleading links and potentially dangerous click-related behaviors. To make our work more usable for end users and developers alike, we decided to implement our checks in a proof-of-concept service that can test a given web page and generate a report describing the bad practices identified in its clickable elements. We believe that such a tool can be useful for end-users interested in validating suspicious websites before visiting them, and in particular for web application developers to discover how they could improve both the usability and the security of their website. Moreover, on top of testing an existing site, our online service also provides a list of guidelines to help developers avoid common mistakes and adhere to the *click contract* described in Section 2. As we cannot expect all web pages to follow the click contract, it is important to introduce a second line of defense to protect the end-users. We implemented a browser extension that could prevent these dangerous side effects.

A proof-of-concept demo of the service, guidelines and extension are publicly accesible at `https://clickbehavior.github.io`.

10 RELATED WORK

Researchers have looked at different ways users are **misled** into performing actions they did not originally intend to perform [9]. For instance, researches from Google analyzed the distribution of fake anti-virus products on the Web [44]. More specific to user clicks, Fratantonio et al. [17] proposed an attack where users are fooled into clicking certain elements while actually clicking on others, Many other works analyzed the specific case of clickjacking [3, 22, 48], where a malicious website tricks the user into clicking on an element of a completely different website by stacking the sites and making the top site invisible.

Redirections are often used for legitimate purposes (e.g., to redirect users from a temporarily moved website), but other times are abused by attacker for malicious reasons. For example, Lu et al. [28] were able to classify different search poisoning campaigns by checking their redirection chains. Stringhini et al. [57] proposed a similar idea to detect malicious webpages. Our work differs in many ways from these approaches, as we check what the possible risks a user may suffer because of obfuscated redirection chains.

The problem of possible **man-in-the-middle** attacks have been extensively analyzed in the Web. Chang et al. [6] screened the integrity and consistency of secure redirections that happen when accessing the main page and login page of domains listed in Alexa. Later, researchers from Google, Cisco, and Mozilla measured the adoption of HTTPS on the web [15]. They conclude that globally most of the browsing activity is secure. Regarding mixed content, Chen et al. [7] investigated the dangers of this type of insecure content. None of the aforementioned studies analyzed how this security and privacy problems is related to the click ecosystem.

Phishing attacks have often been associated to spam emails [16, 21]. Therefore, the majority of the effort to stop this kind of practices was in the early detection of malicious emails [71], or on the detection of phishing pages on the Web [18, 29, 69]. However, we are not aware of any study that tries to identify how common are phishing threats created by insecurely opening new tabs. Our works shows that nearly all the targets opened, either via HTML or directly through JavaScript, suffer from this problem. Even if defenses exist for both cases, they are very rarely implemented.

User **tracking** is an increasingly growing concern that has attracted a considerable amount of attention from researchers and end users [47, 51]. Lerner et al. [26] studied the evolution of tracking over the last 20 years, showing an impressive growth in adoption and complexity. More recently, Sivakorn et al. [56] studied the case of HTTP cookies and the corresponding exposure of private information. On the other hand, we analyzed the concept of undesired cookies that are the consequence of user clicks, and we measured how many of those are insecure.

11 CONCLUSIONS

Using the mouse to click on links and other interactive elements represents the core interaction model of the Web. In this work, we perform the first measurement of click-related behaviors and their associated consequences. We first identified different types of undesired actions that may be triggered when a user clicks on, in principle, harmless elements. In order to assess how widespread these behaviors are on the Internet, we then implemented a crawler, which we used to perform nearly 2.5M clicks on different types of domains of various popularity. Our results show that these dangerous situations are extremely common in all types of domains, making a huge number of users vulnerable to many different possible attacks. Moreover, we offer different possible countermeasures.

ACKNOWLEDGMENTS

This work is partially supported by the Basque Government under a pre-doctoral grant given to Iskander Sanchez-Rola.

REFERENCES

[1] Amazon Web Services. 2019. Alexa Top Sites. https://aws.amazon.com/es/alexa-top-sites/.

[2] Apple. 2019. Manage cookies and website data using Safari. https://support.apple.com/kb/ph21411?locale=en_US.

[3] Marco Balduzzi, Manuel Egele, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. 2010. A Solution for the Automated Detection of Clickjacking Attacks. In *ACM ASIA Computer and Communications Security (ASIACCS)*.

[4] D Basu. 1958. On sampling with and without replacement. *Sankhyā: The Indian Journal of Statistics* 20 (1958).

[5] Mathias Bynens. 2019. About rel=noopener. https://mathiasbynens.github.io/rel-noopener/.

[6] Li Chang, Hsu-Chun Hsiao, Wei Jeng, Tiffany Hyun-Jin Kim, and Wei-Hsi Lin. 2017. Security Implications of Redirection Trail in Popular Websites Worldwide. In *World Wide Web Conference (WWW)*.

[7] Ping Chen, Nick Nikiforakis, Christophe Huygens, and Lieven Desmet. 2015. A Dangerous Mix: Large-scale analysis of mixed-content websites. In *International Journal of Information Security*.

[8] ChromeDevTools. 2019. DevTools Protocol API. https://github.com/ChromeDevTools/debugger-protocol-viewer.

[9] Vacha Dave, Saikat Guha, and Yin Zhang. 2013. ViceROI: Catching Click-Spam in Search Ad Networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[10] Developers Google. 2019. What Is Mixed Content? https://developers.google.com/web/fundamentals/security/prevent-mixed-content/what-is-mixed-content.

[11] Dymo. 2017. Missing Accept_languages in Request for Headless Mode. https://bugs.chromium.org/p/chromium/issues/detail?id=775911.

[12] Serge Egelman and Eyal Peer. 2015. Scaling the Security Wall. Developing a Security Behavior Intentions Scale (SeBIS). In *ACM Conference on Human Factors in Computing Systems (CHI)*.

[13] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[14] European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union* (2016).

[15] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. 2017. Measuring HTTPS adoption on the web. In *USENIX Security Symposium (Sec)*.

[16] Ian Fette, Norman Sadeh, and Anthony Tomasic. 2007. Learning to Detect Phishing Emails. In *World Wide Web Conference (WWW)*.

[17] Yanick Fratantonio, Chenxiong Qian, Simon P Chung, and Wenke Lee. 2017. Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop. In *IEEE Symposium on Security and Privacy (Oakland)*.

[18] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. 2007. A Framework for Detection and Measurement of Phishing Attacks. In *ACM Workshop on Recurring Malcode (WORM)*.

[19] Google. 2019. Opens External Anchors Using rel="noopener". https://developers.google.com/web/tools/lighthouse/audits/noopener.

[20] Google App Maker. 2019. CSS Reference. https://developers.google.com/appmaker/ui/css.

[21] Xiao Han, Nizar Kheir, and Davide Balzarotti. 2016. PhishEye: Live Monitoring of Sandboxed Phishing Kits. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[22] Lin-Shung Huang, Alexander Moshchuk, Helen J Wang, Stuart Schecter, and Collin Jackson. 2012. Clickjacking: Attacks and Defenses. In *USENIX Security Symposium (Sec)*.

[23] Alexandros Kapravelos, Yan Shoshitaishvili, Marco Cova, Christopher Kruegel, and Giovanni Vigna. 2013. Revolver: An Automated Approach to the Detection of Evasive Web-based Malware.. In *USENIX Security Symposium (Sec)*.

[24] Issie Lapowsky. 2018. California Unanimously Passes Historic Privacy Bill. Wired.

[25] Zhulieta Lecheva. [n.d.]. Characterizing the differences of Online Banking User Experience on computer and mobile platforms. *Project Library, AAlborg University* ([n. d.]).

[26] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 2016. Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016.. In *USENIX Security Symposium (Sec)*.

[27] Timothy Libert. 2018. An Automated Approach to Auditing Disclosure of Third-Party Data Collection in Website Privacy Policies. In *World Wide Web Conference (WWW)*.

[28] Long Lu, Roberto Perdisci, and Wenke Lee. 2011. SURF: Detecting and Measuring Search Poisoning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[29] Christian Ludl, Sean McAllister, Engin Kirda, and Christopher Kruegel. 2007. On the Effectiveness of Techniques to Detect Phishing Sites. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.

[30] Kevin Andika Lukita, Maulahikmah Galinium, and James Purnama. 2018. User Experience Analysis of an E-Commerce Website Using User Experience Questionnaire (UEQ) Framework. In *Prosiding Seminar Nasional Pakar*.

[31] William Melicher, Mahmood Sharif, Joshua Tan, Lujo Bauer, Mihai Christodorescu, and Pedro Giovanni Leon. 2016. (Do Not) Track me sometimes: users' contextual preferences for web tracking. In *Privacy Enhancing Technologies Symposium (PETS)*.

[32] Brad Miller, Paul Pearce, Chris Grier, Christian Kreibich, and Vern Paxson. 2011. What's clicking what? techniques and innovations of today's clickbots. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.

[33] Tyler Moore and Benjamin Edelman. 2010. Measuring the Perpetrators and Funders of Typosquatting. In *International Conference on Financial Cryptography and Data Security (FC)*.

[34] Mozilla Foundation. 2019. Cursor - CSS: Cascading Style Sheets. https://developer.mozilla.org/en-US/docs/Web/CSS/cursor.

[35] Mozilla Foundation. 2019. Disable third-party cookies in Firefox to stop some types of tracking by advertisers. https://support.mozilla.org/en-US/kb/disable-third-party-cookies.

[36] Mozilla Foundation. 2019. How do I tell if my connection to a website is secure? https://support.mozilla.org/en-US/kb/how-do-i-tell-if-my-connection-is-secure.

[37] Mozilla Foundation. 2019. Mixed content blocking in Firefox. https://support.mozilla.org/en-US/kb/mixed-content-blocking-firefox.

[38] Mozilla Foundation. 2019. Security/Tracking protection. https://wiki.mozilla.org/Security/Tracking_protection.

[39] Mozilla Foundation. 2019. Window.opener. https://developer.mozilla.org/en-US/docs/Web/API/Window/opener.

[40] OWASP. 2019. Reverse Tabnabbing. https://www.owasp.org/index.php/Reverse_Tabnabbing.

[41] Paul Pearce, Vacha Dave, Chris Grier, Kirill Levchenko, Saikat Guha, Damon McCoy, Vern Paxson, Stefan Savage, and Geoffrey M Voelker. 2014. Characterizing Large-Scale Click Fraud in ZeroAccess. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[42] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, Nagendra Modadugu, et al. 2007. The Ghost in the Browser: Analysis of Web-based Malware. *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)* (2007).

[43] M Zubair Rafique, Tom Van Goethem, Wouter Joosen, Christophe Huygens, and Nick Nikiforakis. 2016. It's Free for a Reason: Exploring the Ecosystem of Free Live Streaming Services. In *Network and Distributed System Security Symposium (NDSS)*.

[44] Moheeb Abu Rajab, Lucas Ballard, Panayiotis Mavrommatis, Niels Provos, and Xin Zhao. 2010. The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.

[45] Aza Raskin. 2019. Tabnabbing: A New Type of Phishing Attack. http://www.azarask.in/blog/post/a-new-type-of-phishing-attack/.

[46] RFC 7235. 2018. Section 3.1: 401 Unauthorized. https://tools.ietf.org/html/rfc7235/.

[47] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and Defending Against Third-Party Tracking on the Web. In *USENIX conference on Networked Systems Design and Implementation (NSDI)*.

[48] Gustav Rydstedt, Elie Bursztein, Dan Boneh, and Collin Jackson. 2010. Busting frame busting: a study of clickjacking vulnerabilities at popular site. *IEEE Oakland Web* 2 (2010).

[49] Iskander Sanchez-Rola, Davide Balzarotti, and Igor Santos. 2017. The Onions Have Eyes: A Comprehensive Structure and Privacy Analysis of Tor Hidden Services. In *World Wide Web Conference (WWW)*.

[50] Iskander Sanchez-Rola and Igor Santos. 2018. Knockin' on Trackers' Door: Large-Scale Automatic Analysis of Web Tracking. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.

[51] Iskander Sanchez-Rola, Xabier Ugarte-Pedrero, Igor Santos, and Pablo G Bringas. 2016. The Web is Watching You: A Comprehensive Review of Web-tracking Techniques and Countermeasures. *Logic Journal of IGPL* 25 (2016).

[52] Evan Sangaline. 2017. Making Chrome Headless Undetectable. https://intoli.com/blog/making-chrome-headless-undetectable/.

[53] Evan Sangaline. 2018. It is *Not* Possible to Detect and Block Chrome Headless. https://intoli.com/blog/not-possible-to-block-chrome-headless/.

[54] Martin Schrepp. 2015. User Experience Questionnaire Handbook. *All you need to know to apply the UEQ successfully in your project* (2015).

[55] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. 2017. Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI)* 4 (2017).

[56] Suphannee Sivakorn, Iasonas Polakis, and Angelos D Keromytis. 2016. The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information. In *IEEE Symposium on Security and Privacy (Oakland)*.

[57] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2013. Shady Paths: Leveraging Surfing Crowds to Detect. Malicious Web Pages. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[58] Symantec. 2017. The Need for Threat Risk Levels in Secure Web Gateways. https://www.symantec.com/content/dam/symantec/docs/white-papers/need-for-threat-tisk-Levels-in-secure-web-gateways-en.pdf.
[59] Symantec. 2017. WebPulse. https://www.symantec.com/content/dam/symantec/docs/white-papers/webpulse-en.pdf.
[60] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. 2014. The Long" Taile" of Typosquatting Domain Names. In *USENIX Security Symposium (Sec)*.
[61] Rick Wash and Emilee Rader. 2015. Too Much Knowledge? Security Beliefs and Protective Behaviors Among United States Internet Users. In *Symposium On Usable Privacy and Security (SOUPS)*.
[62] WebKit. 2018. Intelligent Tracking Prevention 2.0. https://webkit.org/blog/8311/intelligent-tracking-prevention-2-0/.
[63] Craig E Wills and Mihajlo Zeljkovic. 2011. A personalized approach to web privacy: awareness, attitudes and actions. *Information Management & Computer Security* 19 (2011).
[64] Gilbert Wondracek, Thorsten Holz, Christian Platzer, Engin Kirda, and Christopher Kruegel. 2010. Is the Internet for Porn? An Insight Into the Online Adult Industry.. In *Workshop on the Economics of Information Security (WEIS)*.
[65] World Wide Web Consortium. 2005. Uniform Resource Identifier (URI): Generic Syntax. https://tools.ietf.org/html/std66.
[66] World Wide Web Consortium. 2018. CSS Basic User Interface. https://drafts.csswg.org/css-ui-3/.
[67] World Wide Web Consortium. 2018. HTML Specification: Links. https://www.w3.org/TR/html401/struct/links.html.
[68] Haidong Xia and José Carlos Brustoloni. 2005. Hardening Web Browsers Against Man-in-the-Middle and. Eavesdropping Attacks. In *World Wide Web Conference (WWW)*.
[69] Yue Zhang, Jason I Hong, and Lorrie F Cranor. 2007. CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In *World Wide Web Conference (WWW)*.
[70] Leah Zhang-Kennedy, Elias Fares, Sonia Chiasson, and Robert Biddle. 2016. GeoPhisher: The Design and Evaluation of Information Visualizations about Internet Phishing Trends. In *Symposium on Electronic Crime Research (eCrime)*.
[71] Bing Zhou, Yiyu Yao, and Jigang Luo. 2014. Cost-sensitive three-way email spam filtering. *Journal of Intelligent Information Systems* 42 (2014).