Cookie Exfiltration: Hostile or Friendly?

Abdulhai Naqvi
UC Davis

Alicia Siu
The University of California, Davis
aclsiu@ucdavis.edu

Abstract

Recently,third party scripts have become increasingly common in tracking user activity across the web. Previous work in the field has explored how common this is and how trackers collaborate to aggregate infomation.

We take a different approach to see if trackers *sabotage* each other. In this exploratory study, we have found preliminary evidence to indicate that there is some active sabotage between different third party trackers.

ACM Reference Format:
Abdulhai Naqvi and Alicia Siu. 2021. Cookie Exfiltration: Hostile or Friendly?. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Recently, major browsers such as Firefox have begun to block third party cookies for privacy protection[18]. To get around this, third-party domains can provide JavaScript scripts, or trackers, that can be embedded in an HTML <script> tag to aggregate user data across multiple websites and site visits.

This allows third party domains to be passed off as first party domains. While this technique can be used to synchronize information effectively across multiple subdomains for non-tracking purposes, it has emerged as a popular tool to track users in a stateful manner[5][8] as it bypasses the same origin policy (SOP) in web browsers.

We investigate in this paper how often third party domains are engaging in cookie sharing (collaborative) and cookie exfiltration (combative) across Alexa Top 10 Websites [1]. We utilize the OpenWPM framework maintained by the Mozilla Foundation [8] to gather data and D3.js[3] and GraphViz[13] libraries to create interactive visualizations.

2 Background

Recently, consumers have started becoming more concerned about their online privacy. This has resulted in changes in cookie behavior in major browsers with regards to what information third party cookies can access[18]. A third party domain here has been defined as any website that is not directly visited by a user. In our case, we consider different subdomains under the same domain to be third parties to each other.

This has sent web trackers into two distinct but synergistic directions: stateful tracking, where information is stored with the client and stateless tracking, where information is stored with the server[11]. Our work focuses on stateful trackers.

Previously deployed measures to restrict cookie sharing have focused on voluntary usage by tracking websites. These have included the "SameSite" and "HttpOnly"[10] attributes. While useful for overall security, these privacy changes can be bypassed entirely.One approach has been browser and machine fingerprinting, a stateless tracking method[8] .

Another common approach has been to use embedded scripts that can be placed within HTML <script> tags on a webpage[5]. This allows trackers to bypass third party domain limitations entirely through JavaScript.

3 Related Work

Recently, Chen et al. have shone light on this phenomenon[5]. They measured the prevalence of web tracking based on first-party cookies set by third-party scripts or JavaScript code, labelling them as "external cookies", which are used to circumvent browser policies that block third-party cookies.

Their analysis shows that external cookies are widely used and a large number of websites utilize these external cookies allowing third parties to exchange tracking IDs allowing data across time and space.

In earlier work, Roesner et al. investigated different kinds of third-party trackers using detection and classification of both stateful and stateless tracking techniques[14]. They used multiple information vectors, including page fetching, first-party domains and same origin policy, as well as tracking information like using HTTP Referrer headers to communicate information about a visited site to the trackers.

It was found that over 20% of a user's browsing information could be ascertained using these techniques at the time.

Another work, [12] examines cookie exfiltration and identifies instances where first-party cookies are sent to third-parties, but with a focus on CNAME cloaking. CNAME cloaking is a method used to disguise a third-party domain as part of a first-party domain for tracking. The authors analyze

Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and conclude a number of websites perform CNAME redirections, causing sensitive cookies to be leaked to third-party domains. This can be problematic from a privacy standpoint, as personal information and authentication data can be exposed.

With these works in mind, our work aims to examine possible cookie exfiltration and examine the exact usage of first-party cookies by embedded scripts set by third-party cookies. In addition, we have developed an interactive visual applications allowing users to better understand information travel and relevant actors.

4 Methodology

4.1 Gathering Data

We used the open source OpenWPM framework [8], an adapted version of Firefox, for instrumentation. We visited the Alexa Top 10 websites [1]

OpenWPM, by default, collects details on cookies that Javascript interacts with. Since "HttpOnly" cookies cannot be accessed using Javascript and can be excluded, we used this data to populate an initial "Cookies" class. Below is the Python representation of this data structure.

4.1.1 The Cookie Class.

```
class Cookie:
    browserId # The specific OpenWPM
              # browser id used to
              # crawl this website
    cookie_name # name of the cookie
   cookie_host # The website the cookie is set on

    # list of sorted operations associated with
    # each cookie
    list Operations []

    # list of operations that "steal" or
    # "exfiltrate" cookies
    list exfiltrationOperations []
```

We chose three invariants to uniquely identify a cookie. The first is the browser ID of the OpenWPM instance that records the particular cookie.

The second is the name of the cookie. The third is the host that stores the cookie.

In addition, the data structure contains a list of Operations that have been performed on the cookie. Among these, we earmark suspicious operations in a special list, exfiltrationOperations.

4.1.2 The Operations Class.

Each Cookie contains a list of Operations. This data structure is sketched out as follows.

```
class Operation:
    actor # the website or "third party"
          # conducting the operation
    operation # the operation conducted.
              # Options  are "read",
              # "add", "modify"
              # and "delete"
    timestamp # A time stamp from the
              # instrumentation.
              # Used to sort the array of
              # Operations
    cookie_value # The value of the cookie
                 # at
                 # the end of the Operation
    access_method # Whether the operation
                  # was done thru Javascript
                  # or HTTP
    expiration # The expiry date of the
               # Cookie at the end of the
               # Operation
```

4.2 To Catch a Predator

When first parties collaborate with a tracker or third party, it is a collaborative operation.

On the other hand, when a third party changes, reads or deletes a cookie without the consent of the party that sent it, the operation is hostile or adversarial in nature. We call this surreptitious behavior **cookie exfiltration**.

To discover cookie exfiltration, we used a simple criteria. We check whether the original setter for the cookie is the actual first party domain. If it is, then we mark the exfiltration operation as "normal" since it is being carried out with the permission of the first party domain.

If the cookie is not set by the first party domain, then it's original setter is a tracker. Now, things get more interesting. If we see other trackers change this cookie's value, how do we determine whether this is adversarial in nature?

Again, we use a simple methodology. If we find either of the two parties being a substring of each other, then it is considered a "normal" exfiltration. For example, "example.com" reading a cookie from "sub.example.com" will be considered "normal" since they are likely operated by the same real world entity.

Finally, if we see trackers changing or reading cookie data without being the original setters or being related to them, we mark this as suspicious.

If the operation is "read", we mark the operation as "spying". If it's a "modify" or "delete", it becomes a "sabotage" operation.

While the classification is very naive, it gives us a good place to start. One can further refine the various categories. An interesting one would be to see if trackers sabotage data set by each other when the original owner is the actual first party. We leave these questions as an invitation for further research.

4.3 Cookie Nibbler

To better visualize the data collected on cookies and the defined operations performed on each cookie, we created an interactive visualization called the "Cookie Nibbler."

The idea of our visualization is to envision a chain of operations acted upon each cookie by every actor and ordered by timestamp. Upon researching software and tools to generate graph visualizations, our initial approach utilized Graphviz[13], which is an open source graph visualization software.

With the previously processed data, we can use the cookie name and the list of operations to generate a graph of linking nodes, where each cookie expands into a node of operations in that cookie.

To conveniently view this graph, we utilized Dash[2], which is an open-source Python library used for creating reactive, web-based applications. This Dash app made a simple to view visualization that can be seen in a browser. To further enhance the visualization, we attempted to add interactivity with a library called dash-interactive-graphviz[15], which renders the graphviz in a dash component and provides the ability to interact with the graphviz such as zooming, panning, selecting nodes, and adding animations.

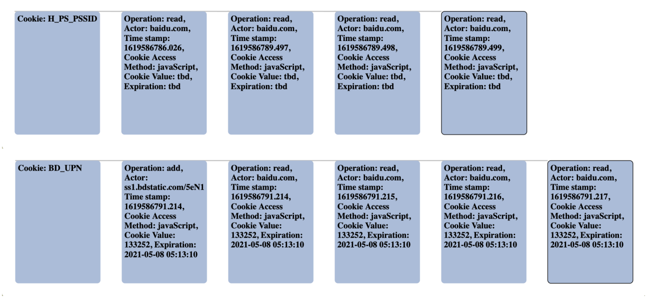
However, due to time constraints, the visualization was not developed further. This initial application can be used as the backbone of a more powerful graphics application as further work.

We used the JavaScript library D3.js[3], which uses HTML, CSS, and SVG. This is used in combination with NodeJS[7], a back-end JavaScript environment for creating web applications.

The application is written in HTML and JavaScript, and runs in a NodeJS environment.

In order to implement the graph visualization with D3.js, further data processing needs to be done as it takes in JSON (JavaScript Object Notation) file. For each cookie, we display a single node. Once a user clicks on each node, the another node is spawned, forming a link. Each node shows an operation performed on the cookie. Within each node, the details of the operation such as the actor and the actual operation, are clearly displayed.

As shown in these example[4][9][16], built a visualization resembling a collapsible tree with multiple root nodes. By manipulating the SVG, interactivity can be added to the visualization, where each node is animated to reveal the next node of the node that is selected. The visualization also includes zooming and dragging abilities so the user can choose to see less or more information. A flourish that was left out due to time constraints was changing the node color based on cookie exfiltration.



Cookie: H_PS_PSSID

Operation: read, Actor: baidu.com, Time stamp: 1619586786.026, Cookie Access Method: javaScript, Cookie Value: tbd, Expiration: tbd

Operation: read, Actor: baidu.com, Time stamp: 1619586789.497, Cookie Access Method: javaScript, Cookie Value: tbd, Expiration: tbd

Operation: read, Actor: baidu.com, Time stamp: 1619586789.498, Cookie Access Method: javaScript, Cookie Value: tbd, Expiration: tbd

Operation: read, Actor: baidu.com, Time stamp: 1619586789.499, Cookie Access Method: javaScript, Cookie Value: tbd, Expiration: tbd

Cookie: BD_UPN

Operation: add, Actor: ss1.bdstatic.com/5eN1 Time stamp: 1619586791.214, Cookie Access Method: javaScript, Cookie Value: 133252, Expiration: 2021-05-08 05:13:10

Operation: read, Actor: baidu.com, Time stamp: 1619586791.214, Cookie Access Method: javaScript, Cookie Value: 133252, Expiration: 2021-05-08 05:13:10

Operation: read, Actor: baidu.com, Time stamp: 1619586791.215, Cookie Access Method: javaScript, Cookie Value: 133252, Expiration: 2021-05-08 05:13:10

Operation: read, Actor: baidu.com, Time stamp: 1619586791.216, Cookie Access Method: javaScript, Cookie Value: 133252, Expiration: 2021-05-08 05:13:10

Operation: read, Actor: baidu.com, Time stamp: 1619586791.217, Cookie Access Method: javaScript, Cookie Value: 133252, Expiration: 2021-05-08 05:13:10

Figure 1. Chain of cookie operations. This illustration displays two cookies with a subset of their corresponding operations.

5 Results

The websites we visited are:

1. google.com
2. youtube.com
3. tmall.com
4. baidu.com
5. qq.com
6. sohu.com
7. facebook.com
8. taobao.com
9. google.com
10. amazon.com

These visits generated 158 cookies from 56 different domains. Of these cookies, 26 had suspicious cookie exfiltration operations.

Of the cookie that were sabotage of exfiltrations, **amazon-adsystem.com** had some 245 exfiltration operations carried out on one of it's cookies.

Upon closer inspection, most of these "sabotages" were actually carried out by other Amazon services and subdomains that were not caught by our simple filters. However, some exfiltrations that were carried out by other websites such as **doubleclick.com** and **adservice.google.com** are hard to explain as non-suspicious without further information of real world relationship information between the two companies.

This example demonstrates the need for more robust filter in any future work carried out to weed out false positives.

With the limitations previously discussed in mind, we found that of the 26 cookies that had successful exfiltration attempts, 17 had a "spy" or "sabotage" operation. Of these, 7 did not have any "spy" operations, only "sabotage" operations.

Properties other than cookie value

So far, we have only looked at the exfiltration of the cookie value. In addition, a cookie has other parameters associated with it as well, such as the SameSite attribute, Expiration date and the HostOnly attribute.

While the cookie value being exfiltrated is the most obvious change, chaning these other parameters can also be a cause of concern.

For example, changing the Expiration date to a date in the past deletes a cookie. Similarly, the SameSite attribute can prevent against Cross Site Resource Forgery (CSRF) attacks [6].

Changing or removing these attributes to a less secure setting can lower the security for these cookies. It also open the door to malicious actors being able to read these cookies for nefarious purposes, like CSRF [6].

Following is an analysis of these secondary cookie properties that might have been exfiltrated or tampered.

SameSite Attribute

The SameSite attribute prevents cookies from being exfiltrated by embedded external resources [17].

In the ten websites, we analyzed earlier, we found 158 cookies from 56 different domains. None of them provided any proof of tampering with the SameSite attribute.

Expiration Date

We found 19 cookies where the expiration date was being changed by third parties who had not originally set the cookie. Some of them were sister domains and subdomains. However, some of these domains such as **sohu.com** had other distinct domains, such as **baidu.com** changing the expirationDate.

HostOnly Attribute

We found 4 cookies that had the HostOnly attribure modified. Again, **baidu.com** was seen modifying cookies from **sohu.com**. It is possible these two companies have a business partnership. Alternatively, they could be trying to actively gain information about **sohu.com**. This merits further research to see how widespread this practice is.

6 Discussion

We only looked at the top 10 websites from Alexa. This was more of an exploratory study than a survey. This was mainly due to time constraints.

Using relatively unoptimized code on our personal machines, we found the running times for a larger set of websites impractical.

Even though our sample size was small, we still found evidence of cookie exfiltration where cookies were being modified or read without the permission of the original domain that set the cookie.

We need more research to confirm whether this is "accidental" or actually malicious.

It is also interesting that some websites engage in changing the HostOnly attribute of cookies they hadn't set. This opens the cookie to be read by domains that didn't set it explicitly. It is hard to imagine this is not malicious if done by two unrelated trackers.

Last but not least, a problem that we have faced is separating truly separate trackers and related domains. While strictly speaking, all third party domains are treated as the same by the browser, one can argue that **abc.com** and **xyz.com** are not engaging in malicious behavior if they are owned by the same parent company and collaborate with each other.

We would need some more information about these trackers and their structure to get a more accurate picture of how much of this behavior is actually malicious.

Additionally, further research is needed with more websites crawled to see how widespread these practices are in the wild.

7 Limitations

A major blind side in our work is the actual relationship between trackers. What we might consider "sabotage" might be collaboration based on a business deal. It is also possible that the two trackers "sabotaging" each other are sister companies held by the same parent company.

Researching these relationships and creating rules based around them would give us a better idea about where actual cookie sabotage or spying is being carried out.

We also haven't made any concessions for "accidental" spying or sabotage. It is very possible that when trackers use "document.cookie" to read cookies, they do not actually use the other information.

However, it is nonetheless a security risk to other trackers who might want the integrity of their cookies to maintain integrity.

8 Future Work

To further the progress made in this work, the cookie visualization can be enhanced by including more features and creating a more appealing and engaging user interface. For example, assigning different colors to each type of node can help to better differentiate the cookie names, and operation types. After identifying suspicious behavior based on the operations performed on the cookies, as well as identifying cookie exfiltration, this can be indicated on the visualization with labels that note which websites contain potentially malicious cookie behavior.

Our criterion for labeling cookie exfiltration operations as "normal" or "spy" or "sabotage" are very simple. While they give us a good baseline to draw on, further refinement is needed to get a better idea about how cookie exfiltration is actually being used in the wild.

One avenue of approach is to check whether multiple actors are sabotaging a cookie's value.

Another interesting method would be to check whether any of the cookies being set or read are hashed copies of existing cookies set by other domains.

It would also be interesting to compare how different domains handle integrity of their cookies. Hashing or encrypting cookie data may provide an integrity check, alerting the cookie reader to a data violation.

In addition to enhancing the cookie visualization to convey more information, a helpful tool that can be created is to utilize machine learning techniques to identify and classify websites in which third party scripts access or change first party scripts. The data gathered and processed in our work can be used as a training set for the machine learning algorithm. Possible classification machine learning algorithms that can be taken into consideration are logistic regression and naive bayes classifiers. Performing experiments on different algorithms will determine which machine learning algorithms result in the best performance in a short amount of time.

9 Conclusion

We have found evidence of cookie exfiltration and tampering, sometimes collaborative, and sometimes not, on ten of the most popular websites from Alexa. It is an promising area of research. Other works, for example Chen [5], look at third party trackers that are embedded to get around the Same origin policy on web browsers.

However, more work needs to be done to see how trackers are interacting with each other beyong strict collaboration. We have preliminary evidence that there is some sabotage and snooping of other third party cookies.

10 Contributions

Alicia Siu's main contribution to this project is visualizing the cookies and creating the visualization for the cookie operation. She researched, tested and implemented graph visualizations using different software tools and libraries to create an interactive visualization that the team envisioned. She helped with initial analysis of the crawled data in the database. She also contributed to the project proposal and project powerpoint presentations, and contributed largely in writing the mid-point progress report including research on related work. In this paper, she wrote sections on related work, measurement description on the cookie visualization, future work, and bibliography.

Abdulhai contributed to the data structures and overall architecture.

The code can be found here: https://github.com/ahnaqvi/cookie_exfiltration/

References

[1] [n.d.]. The top 500 sites on the web. https://www.alexa.com. Accessed: 2021-06-08.

[2] 2019. *Dash documentation*. Retrieved June 7, 2021 from https://dash.plotly.com/

[3] Mike Bostock. 2012. D3.js - Data-Driven Documents. http://d3js.org/

[4] Mike Bostock. 2020. *Collapsible Tree*. Retrieved June 7, 2021 from https://bl.ocks.org/mbostock/4339083

[5] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. 2021. Cookie Swap Party: Abusing First-Party Cookies for Web Tracking. In *Proceedings of the Web Conference 2021*. 2117–2129.

[6] Coram. [n.d.]. Cross-Site Request ForgeryChallenges and Solutions. https://www.osti.gov/servlets/purl/1639993. Accessed: 2021-06-08.

[7] Ryan Dahl. 2009. *Node.js*. Retrieved June 7, 2021 from https://nodejs.org/en/

[8] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of ACM CCS 2016*.

[9] ialarmedalien. 2018. *Fake Multi-rooted Tree*. Retrieved June 7, 2021 from http://bl.ocks.org/ialarmedalien/c92a58f2fee695c3931c1b6e30540d98

[10] MDN. 2021. *Using HTTP cookies*. Retrieved June 7, 2021 from https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

[11] Redhat. 2021. *Stateful-vs-stateless*. Retrieved June 7, 2021 from https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless

[12] Tongwei Ren, Alexander Wittman, Lorenzo De Carli, and Drew Davidson. 2021. An Analysis of First-Party Cookie Exfiltration due to CNAME Redirections. (2021).

[13] ATT Labs Research. 1991. *Graphviz*. Retrieved June 7, 2021 from https://graphviz.org/

[14] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and defending against third-party tracking on the web. In *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*. 155–168.

[15] James Saunders. 2021. *dash-interactive-graphviz*. Retrieved June 7, 2021 from https://pypi.org/project/dash-interactive-graphviz/

[16] swayvil. 2021. *Collapsing Tree with Boxes*. Retrieved June 7, 2021 from https://bl.ocks.org/swayvil/b86f8d4941bdfcbfff8f69619cd2f460

[17] West. [n.d.]. Same-site Cookies. https://datatracker.ietf.org/doc/html/draft-west-first-party-cookies-07. Accessed: 2021-06-08.

[18] Wood. 2019. *Today's Firefox Blocks Third-Party Tracking Cookies and Cryptomining by Default*. Retrieved June 7, 2021 from https://blog.mozilla.org/en/products/firefox/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/