Pixel Wayback

Abstract

A clear and well-documented LaTeXdocument is presented as an article formatted for publication by ACM in a conference proceed- ings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

CCS Concepts

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Keywords

Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:

. 2018. Pixel Wayback. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

2 Background

2.1 What is the Meta Pixel?

2.1.1 ***The Concept of Meta Pixel.*** The Meta Pixel is a snippet of JavaScript code that enables the tracking of visitor activity on a website. It operates by loading a small library of functions, which are used to track actions performed by visitors on the website, referred to as *events*. These events are defined by the pixel's administrator, who has access to the pixel instance and determines the specific actions to be monitored, known as *conversions* [17]. With Meta Pixel integrated, a website can measure and optimize the effectiveness of its advertising campaigns by targeting specific audiences based on their interactions, such as interest in certain products or services. Moreover, the visitor's activities can be sent to Meta, even if the visitor does not have an account on any of Meta's platforms, such as Instagram or Facebook.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06

https://doi.org/XXXXXXX.XXXXXXX

2.1.2 ***Evolution of the Meta Pixel.*** Over the years, Meta has refined its user-tracking capabilities, extending its tracking abilities to third-party websites with the help of pixels. In 2007, Facebook introduced Beacon, an early system designed to track user activities on partner websites. Beacon operated by embedding a small piece of code on these websites, allowing Facebook to collect data on users' actions on third-party websites and share them on their Facebook profiles[15] [21]. However, the lack of explicit user consent triggered widespread privacy concerns, leading to the program's discontinuation in 2009[3]. In 2010, during the f8 conference, Facebook allowed websites to implement a Like button on their own pages, enabling users to share content from those sites with their connections on Facebook. This feature not only allowed users to express approval but also facilitated Facebook's ability to track user activities across these sites. However, the feature has faced criticism due to privacy concerns, as it enabled Facebook to trace user actions back to their identity on the platform [20]. Later, in 2013, Facebook introduced conversion and audience pixels. The conversion pixel was used to signal events that happened while the user was browsing a website by setting up to 5 standard events, including *Checkouts, Add to Cart, Key Page Views, Leads, Registrations, and Checkouts*. The audience pixel, on the other hand, was primarily used to build custom audiences based on the activity of users on websites, allowing advertisers to retarget users who have visited the site or specific pages, tailoring future ads to these audiences [14].

With a major update in 2015, Facebook announced the Facebook Pixel (now called the Meta Pixel), which unified the capabilities of both conversion and audience pixels. The new edition offered faster loading times, up to 9 standard events, support for configuring custom events, and allowed sharing of the same pixel across multiple websites to ease collaborations with agencies or other businesses[7] [14]. The Facebook Pixel thus enhanced Facebook's ability to gather detailed insights into users' online actions while simplifying the setup process for advertisers.

2.2 How does Meta Pixel track users?

2.2.1 ***What information does Meta collect?*** Meta utilizes *event-driven* tracking on websites, meaning that the webpage administrator must define the specific behaviors or actions they wish to monitor. These defined events are then sent to Meta, enabling the platform to optimize the retargeting of advertisements for the webpage. The current version of the Meta Pixel allows administrators to configure any of the 18 standard events provided by Meta, attach parameters to these events (if permitted), or define custom events with custom parameters tailored to their specific needs. Standard events include predefined actions such as `PageView`, `AddToCart`, and `Purchase`, while custom events allow website managers to

define and monitor user interactions tailored to their business needs. These events can be linked to specific triggers, such as button clicks or scrolling, enabling precise tracking of user behavior and conversion rates. The configuration for these events is typically managed via the *Events Manager* on `business.facebook.com` [1].

2.2.2 ***How does Meta identify users?*** Pixel administrators also have the ability to control the data sent to Meta's servers by configuring settings in the *Events Manager* desk at https://business.facebook.com. From the desk, administrators can enable *First Party Cookies*, *Automatic Advanced Matching*, and *Core Setup*.

(1) **First Party Cookies**
Enabling the use of first-party cookies allows the pixel to set up the `_fbc` and `_fbp` cookies to be sent along with every event from the website [8]. Prior work has shown that these cookies, in combination, can allow Meta to link the identity of the visitor to the website with their actual profile on Meta's platforms, enhancing ad retargeting [13].
(2) **Automatic Advanced Matching**
Administrators can also configure Automatic Advanced Matching, which, if enabled, automatically sends hashed versions of the visitor's *email, first name, last name, phone number, gender, zip code, city, state, country, date of birth, and external ID* to Meta if provided on the website. This helps attribute more conversions to the business's Meta advertisements. Administrators can configure filters to include or exclude specific combinations of these hashed identifiers [2] [9].
(3) **Core Setup**
This option enables pixel administrators to restrict the transmission of custom parameters or any URL components beyond the domain to Meta, thereby preventing the sharing of information that may violate the Meta Business Tools Terms. In cases of such violations, Meta may itself enable the Core Setup configuration, in which case pixel administrators are not allowed to disable the configuration [10] .

2.3 Meta Pixel Configuration File

In Figure 1, we illustrate the chain of actions which load the configuration file. When a user visits a website, the base function `fbq`, configured with a unique Pixel ID, is initialized, and a library of functions called *fbevents.js* [4] (previously *fbd.js [5]*) is loaded. The *fbevents.js* library subsequently loads another configuration file specific to the Pixel ID. This configuration file defines how this specific Pixel instance tracks events and transmits data to Meta's servers according to the types of events set up and the permissions configured in the Event Manager desk. Interpreting the code in these configuration files can thus provide valuable insights into the potential transmission of sensitive user information through website interactions.

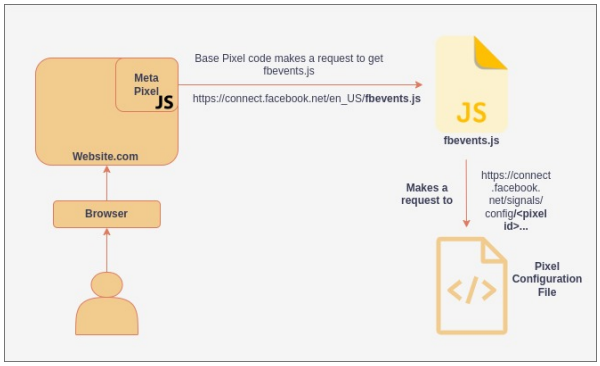

Figure 1: Overview of the chain of events that load the pixel's configuration file.

Based on our examination, much of the configuration code appears to be minified and obfuscated, which makes it challenging to interpret directly. Since Meta allows managers to control the types of events and data sent to its servers via the pixel through the Events Manager desk, we conduct a differential analysis of the configuration code. This involves modifying permissions, enabling or disabling specific events, modifying event parameters, and applying restrictions through the platform as permitted to observe how these changes are reflected in the configuration source code.

It was observed in the configuration code that a segment towards the end, within the `fbq.registerPlugin(...)` function, displayed structured patterns related to changes in pixel settings. The code was found to be relatively interpretable, with configurations such as *FirstPartyCookies* being "opted in," the plugins required for this configuration being loaded, and the configuration being further modified. Therefore, we focus our longitudinal analysis on this segment of the code to track and relay the tracking behaviors of the respective website.

2.4 Functional Segmentation

The aforementioned segment of code, which we call the configuration segment, was modularly formed with the five following sets of functions:

(1) **`fbq.registerPlugin(. . . )`**
- This function wraps the entire configuration segment, containing all the remaining functions within itself.

(2) **`fbq.loadPlugin(<module name>)`**
- This function loads modules associated with specific configurations, such as "First Party Cookies"
- Example:

```
fbq.loadPlugin("cookie");
```

(3) **`instance.optIn(<Pixel ID>, <configuration name>, <bool>)`**
- If true, opts in the respective configuration for this pixel instance.
- Example:

```
instance.optIn("1752529721656045", "
    FirstPartyCookies", true);
```

(4) **config.set(<Pixel ID>, <configuration name>, <configuration setup in JSON>)**
- Allows configuring the configuration at a higher granularity.
- Example:

```
config.set("1494993704116832", "cookie
    ", {
    "fbcParamsConfig": {
        "params": [{
            "prefix": "",
            "query": "fbclid",
            "ebp_path": "clickID"
        }, {
            "prefix": "aem",
            "query": "aem",
            "ebp_path": "aem"
        }]
    },
    "enableFbcParamSplit": false
});
```

(5) **fbq.set(<configuration name>, <Pixel ID>, <list>)**
- Used for listing types of events and their associated behaviors (need improvement).
- Example:

```
fbq.set("iwlExtractors",
    "1752529721656045", []);
```

The configuration segment was parsed into these modules, capturing all the types of configurations they addressed, their configuration setup in the case of `config.set()`, and the list of event behaviors in the case of `fbq.set()`.

3 Methodology

In this section, we organize the methodology of our work organized as follows. We first explain the selection of websites and the curation of their snapshots longitudinally backward (Section 3.1), from which we identify and extract the Meta pixel IDs (Section 3.2). Next, we crawl Meta Pixel configurations across different points in time (Section 3.3) to perform a differential analysis of these configurations (Section 3.4).

3.1 Website Selection

Our goal is to investigate the tracking behaviors of Meta Pixel on websites over time. Using hospital and children's websites as a case study, we aim to investigate how websites track visitor information and analyze their compliance with relevant regulations.

3.1.1 ***Hospital and Children's Website Selection.*** Hospital websites were curated using a dataset of 5685 registered hospitals in the United States from AHA DataQuery [11]. For the children's websites, we use a list of 2006 websites detected and used in a recent work [18]. Next, to ensure consistency in the collected data, we retain only the homepages of all websites present in the lists by removing any path parameters or sub-page addresses. Moreover, we pre-processed the resulting websites by crawling all live versions using a Selenium-controlled Chrome browser, filtering out non-functional sites, and removing any duplicates. As an additional measure for children's websites, we also analyze the TLD to filter out sites not based in the USA, ensuring we focus on those subject to U.S. regulations. For example, a TLD like '.tr' indicates a site is based in Turkey, and so it was removed. This left us with 1,039 children's websites and 3,272 hospital websites, each distinct, for further analysis."

3.1.2 ***Selection of control websites.*** To facilitate the comparison of tracking behaviors of children and hospital websites, we use the Tranco List of top-ranked websites [16] to form a set of control websites. The list version ID: V93YN, dated 13th September 2024, which ranks the top 1 million websites, was employed. Due to the strict crawling limits of Wayback, experiments were limited to the top 10,000 websites, ensuring a manageable yet comprehensive sample. Since these sites cover a broad spectrum of topics without being confined to any specific category, they serve as a diverse benchmark for understanding general tracking practices.

3.2 Crawling Website Snapshots

Longitudinally analyzing the tracking behaviors of a website requires collecting a pixel's configuration files over time. We conduct a retrospective measurement study to analyze how a website's tracking behavior evolves over time. To gather historical snapshots of our selected websites and then their respective pixel configuration files, we use the Wayback Machine. The Wayback Machine has periodically archived popular websites and their resources (e.g., scripts, images) since 1996 and has already archived more than 600 billion web pages thus far[12]. Gathering historical snapshots of a pixel's configuration files requires access to its unique pixel ID, as well as a valid timestamp of the historical snapshot. We first begin by crawling a website longitudinally through Wayback to detect and extract any valid pixel IDs since a website may use multiple different pixels over time and use them to construct valid URLs for historical pixel configuration snapshots.

We used the CDX Server API to collect historical snapshot records of websites available on Internet Archive's Wayback Machine, enabling us to crawl selected websites longitudinally backward in time. The CDX records provide metadata about available snapshots, including timestamps, which allow us to construct appropriate Wayback Machine URLs for each snapshot [6].

3.2.1 ***Wayback Snapshot URL Generation.*** The API was queried in batches of up to 100,000 records per request for each website, retrieving snapshot records backward in time until 2017. For the majority of websites, a single request was sufficient to retrieve all available records. By limiting the

number of CDX API retries to five, snapshot timestamps obtained from the records were used to generate corresponding archive.org URLs. For example, the URL for the Wayback's snapshot of ***https://www.facebook.com/*** taken on September 1, 2024, is ***https://web.archive.org/web/20240901000408/https://www.facebook.com/***.

3.2.2 ***Crawling Website Snapshots***. Due to Wayback's crawling limits, we capped website snapshots to twice per year (on January 1st and July 1st) for hospital and children's websites and once per month for control websites, as it is unlikely that many different Meta Pixels would be used over a short period. Each archived website was accessed using a clean instance of Chromium version 114.0.5735.198, controlled via Selenium to enable automated crawling [22]. We use Selenium's default page load strategy, ensuring that each webpage is fully loaded during the crawl, allowing dynamic elements to be rendered in the HTML [19]. Moreover, given that automated crawling can face network errors, disconnections, unavailability of web servers, etc., we visit each website ten times.

3.3 Identification and Extraction of Meta Pixel IDs

The Base Pixel code is a small JavaScript segment that serves as the 'initiator' for Meta Pixel's overall functionality. The base function, `fbq()`, is initialized with a unique Pixel ID, which triggers the retrieval of a configuration file from *https://connect.facebook.net/signals/config/<Pixel-ID>/*. This file configures the pixel with settings specific to the instance managed by the pixel manager.

To extract the unique Pixel ID from a website's Meta Pixel, we analyze the HTML of its snapshots. Our analysis reveals that the Pixel ID can be found either in the script tag that loads the configuration file or in the initialization of the `fbq()` function. For instance, a website like ***example.com*** might include the Pixel ID 1234 as ***<scriptsrc="https://example.com/signals/config/1234?v=..."async=""></script>*** or within the initialization ***fbq("init", 1234)***, or both. We utilize regular expressions to extract all Pixel IDs based on these patterns.

3.4 Crawling Meta Pixel Configuration Files

In this section, we explain how a differential analysis was conducted to facilitate our understanding of the code.

Pixel IDs extracted from website snapshots were used to crawl all snapshots of the respective pixel's configuration file longitudinally backward. Similar to Wayback Snapshot URL Generation, the CDX Server API was used to obtain records for all existing snapshots of a configuration file by querying for the URL https://connect.facebook.net/signals/config/<PixelID> as a prefix. All CDX records for the given URL prefix were fetched through the CDX API to form appropriate URLs on *archive.org*, as was done for crawling website snapshots. For instance, a valid Wayback snapshot of ***example.com*** integrated with a pixel whose ID is 1234 taken on August 1, 2024, could have the URL ***https://web.archive.org/web/20240801000502/https://connect.facebook.net/signals/config/1234...***.

Although Selenium was used to render JavaScript and extract Pixel IDs from website snapshots, configuration file snapshots from Wayback were downloaded directly using the requests library, as they are static JavaScript files that do not require rendering. To minimize redundant downloads, we only downloaded the latest available configuration snapshot for each day when multiple snapshots were present. This approach created a longitudinal dataset of Pixel configurations, capturing changes over time from Wayback.

4 Differential Analysis Approach

To facilitate the analysis of Pixel Configuration files, we conduct a differential analysis whereby available configurations in the *Events Manager Desk* were modified, and any differences in the configuration file were noted. In this section, we provide our approach to conducting a differential analysis based on the available configurations in the desk and our findings.

4.1 Environment Setup and Control Configuration

We begin by setting up a Facebook Business Account, through which a Meta Pixel is set up. The base code provided by Meta for installing Pixel was manually added to the head element of a test website.

For a starting reference point, a control configuration was set up to compare all observed changes later. In the settings of the event manager desk, the following setup corresponds to the control configuration:

- Data Restrictions, Core Setup: **Off**
- Automatic Website Matching: **Off**
- First-Party cookies: **Off**
- Custom events: **None**
- Standard events: **None**
- Microdata: **Not Setup**
- Track Events Automatically Without Code: **Off**
- Extend Attribution Uploads: **Not Checked**
- Allow Historical Conversion Uploads: **Not Checked**
- Domains/Subdomains in block or allow list: **None**
- Automatic event logging for the Facebook SDK: **Unspecified**

4.2 Treatment Configurations

Next, we modified the configurations in the control setup and documented the observed changes in the pixel's configuration file. **Table 1** summarizes which additional plugins are loaded, which instances are opted in, and which configuration is further configured. We provide below a brief description of the configurations and some details of the changes observed.

4.2.1 Core Setup. Core Setup is enabled from the *Events Manager Desk* as shown in Figure 2.

Feature	loadPlugin Added	instance.Optin Added	config.set Added/-Modified	fbq.set Added/Modified
Core Setup	protecteddatamote, ccruleevaluator	protectedDataMode, CCRuleEvaluator.	protectedDataMode, ccRuleEvaluator, standardParamCheck, inferredEvents	No Change
Automatic Matching	inferredevents, identity.	AutomaticMatching	automaticMatching	No change.
First-Party Cookies	cookie, browserproperties	FirstPartyCookies, BrowserProperties	cookie, browserProperties	No Change
Event Setup	iwlparameters, estruleengine.	ESTRuleEngine, IWLParameters	inferredEvents	iwlExtractors
microdata	jsonld$_{m}icrodata$.	MicrodataJSONLd, Microdaaz	inferredEvents	iwlExtractors

Table 1: Modifications to Meta Pixel Configuration Code

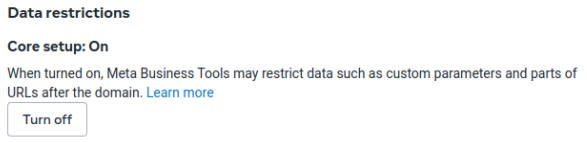

Figure 2: Enabling Core Setup as a treatment to the control configuration

Enabling Core Setup restricts the transmission of custom parameters and anything in a URL following the domain. Firstly, it was observed that enabling *Core Setup* changed the value of the `disableRestrictedData` key within the *inferredEvents* configuration from `false` to `true`.

```
config.set("25826907853621873","inferredEvents
    ",{"buttonSelector":null,
    "disableRestrictedData":true
});
```

Secondly, the configurations for ***protectedDataMode*** and ***ccRuleEvaluator*** were opted in, their respective modules were loaded, and were further configured as follows:

```
    config.set("25826907853621873","
        protectedDataMode",{
            "standardParams":{
                {
                    "lead_event_source":true,
                    "net_revenue":true,
                    "predicted_ltv":true,
                    "product_catalog_id":true,
                    ...
                    "trim":true,
                    "user_bucket":true,
                    "value":true,
                    "vin":true,"year":true
                }});

    config.set("25826907853621873", "
        ccRuleEvaluator", {
        "ccRules": [],
        "wcaRules": [{
            "rule": {
                "and": [{
                    "url": {
                        "i_contains": ""}}]
                    },
            "id": "9190481024316480"
        }],
        "blacklistedIframeReferrers": {
            "google": true
        }
    });

    fbq.loadPlugin("protecteddatamode");
    fbq.loadPlugin("ccruleevaluator");

    instance.optIn("25826907853621873","
        ProtectedDataMode",true);
    instance.optIn("25826907853621873","
        CCRuleEvaluator",true);
```

4.2.2 Automatic Matching. Automatic Matching is enabled, and all customer information parameters are configured as a treatment, as shown in Figure 3.

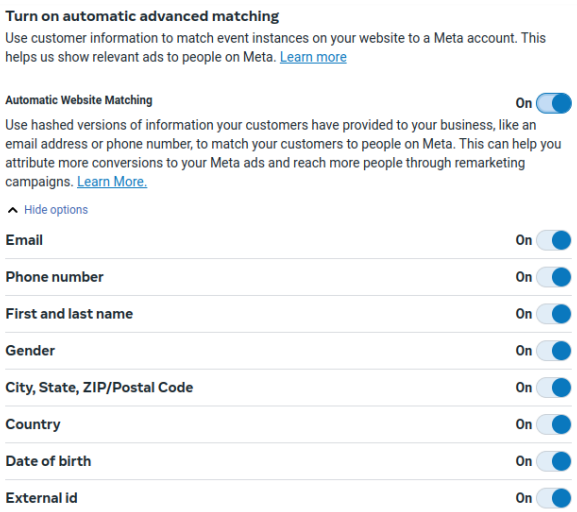

Figure 3: Enabling advanced automatic matching as a treatment to the control configuration

Enabling Automatic Matching, as shown above, brings the following changes to the configuration file. The key *selectedMatchKeys* contains the customer information parameters as were configured through the desk:

```
config.set("25826907853621873","
    automaticMatching",{
"selectedMatchKeys":["em","fn","ln","ge","
    ph","ct","st","zp","db","country","
    external_id"]
});

fbq.loadPlugin("inferredevents");
fbq.loadPlugin("identity");

instance.optIn("25826907853621873","
    AutomaticMatching",true);
```

4.2.3 First Party Cookies. The First Party Cookies configuration was enabled from the desk, as shown in Figure 4.

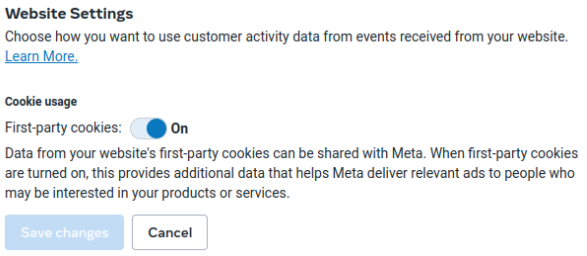

Figure 4: Enabling first-party cookies as a treatment to the control configuration

Enabling the configuration allows Meta to send the `fbq` and `fbc` cookies along with events as parameters. Both cookie and browse property configurations were observed to be configured as follows:

```
config.set("25826907853621873", "
    browserProperties", {
    "delayInMs": 200,
    "enableEventSuppression": true,
    "enableBackupTimeout": true,
    "fbcParamsConfig": {
        "params": [{
            "prefix": "",
            "query": "fbclid",
            "ebp_path": "clickID"
        }, {
            "prefix": "aem",
            "query": "aem",
            "ebp_path": "aem"
        }]
    },
    "enableFbcParamSplit": false
});

config.set("25826907853621873", "cookie",
    {
    "fbcParamsConfig": {
        "params": [{
            "prefix": "",
            "query": "fbclid",
            "ebp_path": "clickID"
        }, {
            "prefix": "aem",
            "query": "aem",
            "ebp_path": "aem"
        }]
    },
    "enableFbcParamSplit": false
});

fbq.loadPlugin("cookie");
fbq.loadPlugin("browserproperties");

instance.optIn("2582690785362187 3","
    FirstPartyCookies",true);
instance.optIn("25826907853621873","
    BrowserProperties",true);
```

4.2.4 Event Setup. Meta allows administrators to configure standard events through three methods:

(1) Manually adding code to the website using the *fbq('track', <Standard Event>)* function,
(2) Via integrations with partner platforms such as Shopify, or
(3) Automatically through the *Event Setup Tool* at *Events Manager Desk*.

The *Event Setup Tool* can detect and suggest events based on button text that matches standard pixel event names. It also

enables administrators to configure value and currency parameters for these events without adding additional code. Here, the value represents the conversion amount in the respective currency of the event that can be tracked from the page, such a basket total[?]. Additionally, administrators can configure custom events by manually adding code and optionally sending custom parameters as needed.

As part of our differential analysis, we perform the following treatments:

(1) Use the *Event Setup Tool* to configure standard events without any value and currency parameters.
   - A Contact event was set up to trigger when the URL of the website was equal to *https://react-portfolio-alpha-nine-57.vercel.app/contact*. This resulted in the following code being added to the configuration code:

```
fbq.loadPlugin("iwlparameters");
fbq.loadPlugin("estruleengine");

instance.optIn("25826907853621873","
    ESTRuleEngine",true);
instance.optIn("25826907853621873","
    IWLParameters",true);

fbq.set("iwlExtractors
    ","25826907853621873",[]);
fbq.set("estRules",
    "25826907853621873", [{
            "condition": {
                "type": 1,
                "conditions": [{
                    "targetType": 2,
                    "extractor": 1,
                    "operator": 2,
                    "action": 2,
                    "value": "https
                        :\/\/react-
                        portfolio-alpha
                        -nine-57.vercel
                        .app\/contact"
                }]
            },
            "derived_event_name": "Lead
                ",
            "transformations": [1],
            "rule_status": "ACTIVE",
            "rule_id":
                "558450303610271"
        }]);
```

   - A Lead Event was set, where the following element on the test website was tracked: t<button class="btn ac_btn" type="submit" style="cursor: none;">Send</button>. This resulted in the following code being added to the configuration where derived_event_name is the name of the standard event set up, and the value is the innerHTML of the element being tracked:

```
fbq.loadPlugin("iwlparameters");
fbq.loadPlugin("estruleengine");

instance.optIn("25826907853621873","
    ESTRuleEngine",true);
instance.optIn("25826907853621873","
    IWLParameters",true);

fbq.set("iwlExtractors
    ","25826907853621873",[]);
fbq.set("estRules",
    "25826907853621873", [{
            "condition": {
                "type": 1,
                "conditions": [{
                    "targetType": 1,
                    "extractor": 2,
                    "operator": 2,
                    "action": 1,
                    "value": "send"}]},
            "derived_event_name": "Lead
                ",
            "transformations": [1],
            "rule_status": "ACTIVE",
            "rule_id":
                "1113739117040193"
        }]);
```

Moreover, the buttonSelector key in the *inferredevents* configuration changed from **null** to **extended** as shown:

```
config.set("25826907853621873","
    inferredEvents",{
    "buttonSelector":"extended",
    "disableRestrictedData":false"
});
```

Note that between the two events set using URL and button, the targetType key inside the *estRules* configuration changes from 2 to 1, suggesting that 2 represents the target event tracking a URL while 1 is the case where the event tracked a button. Similarly, the value of action changes from 1 to 2 as well, while the value of *extractor* key changes from 2 to 1 as we track a button instead of a URL.

(2) Use the *Event Setup Tool* to configure Standard events with value or currency parameters A lead event was tracked using the same button as above, except a phone number was tracked as the value of the currency being in dollars. The same configuration difference as above was above except list in the argument of *fbq.set("iwlExractors, <Pixel ID>, [])* was now populated as follows:

```
    fbq.set("iwlExtractors",
        "25826907853621873", [{
            "domain_uri": "https:\/\/
                react-portfolio-alpha-
                nine-57.vercel.app\/
                contact",
            "event_type": "Lead",
            "extractor_config": {
                "parameter_selectors": [{
```

```
                    "parameter_type": "
                        value",
                    "selector": "closest
                        (.sec_sp.row)
                        ADDRESS P"
                }]
            },
            "extractor_type": "CSS",
            "id": "379257155213763"
        }, {
            "domain_uri": "https:\/\/
                react-portfolio-alpha-
                nine-57.vercel.app\/
                contact",
            "event_type": "Lead",
            "extractor_config": {
                "parameter_type": "
                    currency",
                "value": "USD"
            },
            "extractor_type": "
                CONSTANT_VALUE",
            "id": "916462507217219"
        }]);
```

(3) Manually add standard and custom events through code, with and without events
In the HTML of the test website, the following functions were placed to fire upon a user writing to an input field: *fbq('track', <Standard Event>)*, *fbq('trackCustom', <Custom Event>)*, *fbq('track', <Standard Event>, <Parameters>)*, and *fbq('trackCustom', <Custom Event>), <Parameters>*
In every case, the configuration file was not modified in any way, suggesting that only events setup through the *Event Setup Tool* are reflected in the configuration file

4.2.5 Microdata. Microdata is an HTML specification used to nest metadata within existing content on web pages. It uses a supporting vocabulary to describe an item and employs name-value pairs to assign values to its properties. We utilized `JSON-LD` to add a product object whose data is sent with certain events that are also configured. This was implemented as code within the `<head>` element of the webpage. Due to this, the following code was added to the configuration file:

```
config.set("25826907853621873","microdata",{"
    waitTimeMs":1, "disableMicrodataEvent":
    true,"enablePageHash":true });

fbq.loadPlugin("jsonld_microdata");
fbq.loadPlugin("microdata");

instance.optIn("25826907853621873","
    MicrodataJsonLd",true);
instance.optIn("25826907853621873","Microdata
    ",true);
```

4.2.6 Track Events Automatically Without Code. The configuration available in Events Manager settings was turned on as shown in Figure 5

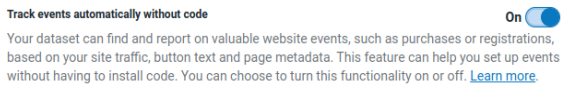

Figure 5: Enabling Track Events Without Code as a treatment to the control configuration

No difference in the configuration file was observed after enabling this configuration.

4.2.7 Extend Attribution Uploads. The configuration available in Events Manager settings was turned on as shown in Figure 6

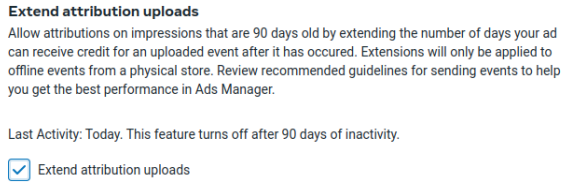

Figure 6: Enabling Extend Attribution Uploads as a treatment to the control configuration

No difference in the configuration file was observed after enabling this configuration.

4.2.8 Allow Historical Conversion Uploads. The configuration available in Events Manager settings was turned on as shown in Figure 7

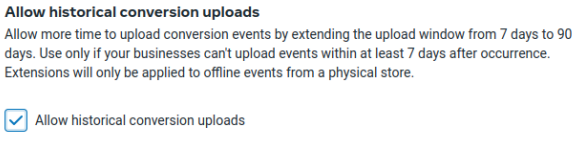

Figure 7: Enabling Track Events Without Code as a treatment to the control configuration

No difference in the configuration file was observed after enabling this configuration.

4.2.9 Adding domains to Allow/Block list. *facebook.com* was added first to an Allow list, as shown in Figure 8.

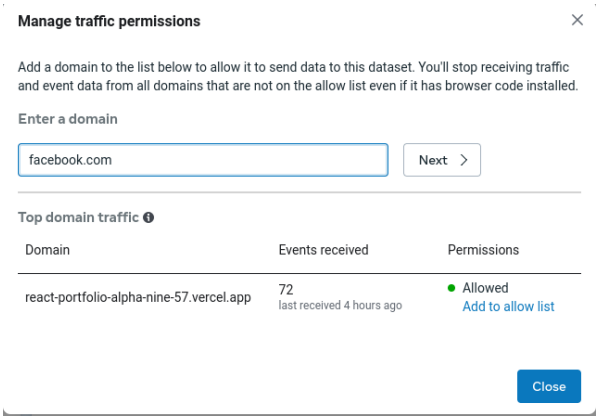

Figure 8: Adding facebook.com to the the allow list

Similarly, *facebook.com* was then added to a block list after being removed from the allow list. In either cases, no difference in the configuration file was observed.

4.2.10 Automatic event logging for the Facebook SDK. The configuration was turned on through the Events Manager settings as shown in Figure 9.

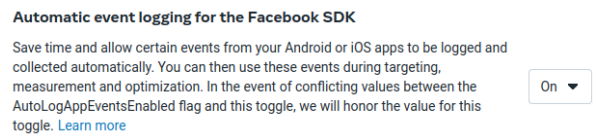

Figure 9: Turning on automatic event logging

5 Case Studies

5.1 Healthcare Websites

5.2 Kids Websites

6 Discussion

7 Conclusion

References

[1] [n. d.]. https://www.facebook.com/business/help/402791146561655?id=1205376682832142
[2] [n. d.]. https://www.facebook.com/business/help/611774685654668?id=1205376682832142
[3] 2009. https://www.cbc.ca/news/science/facebook-shuts-down-beacon-marketing-tool-1.832698
[4] 2017. https://connect.facebook.net/en_US/fbevents.js
[5] 2017. https://connect.facebook.net/en_US/fbds.js
[6] 2018. https://archive.org/developers/wayback-cdx-server.html
[7] 2022. https://developers.facebook.com/ads/blog/post/2015/06/10/upgrades-to-conversion-tracking/
[8] 2022. https://www.facebook.com/business/help/471978536642445?id=1205376682832142
[9] 2022. https://developers.facebook.com/docs/meta-pixel/advanced/advanced-matching/
[10] 2022. https://www.facebook.com/business/help/124742407297678
[11] 2024. https://www.ahadata.com/aha-dataquery
[12] Internet Archive. 2019. Wayback Machine. https://web.archive.org/
[13] Paschalis Bekos, Panagiotis Papadopoulos, Evangelos P. Markatos, and Nicolas Kourtellis. 2023. The Hitchhiker's Guide to Facebook Web Tracking with Invisible Pixels and Click IDs. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23)*. Association for Computing Machinery, New York, NY, USA, 2132–2143. https://doi.org/10.1145/3543507.3583311
[14] Antonio Calero. 2015. How to Use Facebook's Upgraded Website Custom Audience Pixel - Jon Loomer Digital. https://www.jonloomer.com/facebook-upgraded-pixel/?fbclid=IwY2xjawFxqvFleHRuA2FlbQIxMQABHS3zxN3BRZrdInNVCTIOesdMGe30HkL3N85qCEtMovBMj_1wvXgyLgacyA_aem_Nmp5P0YfV4lW8-Ml-yklTQ
[15] Dave. 2024. Facebook Beacon Privacy Settings for External Websites (ex: BustedTees.com). https://500hats.typepad.com/500blogs/2007/11/facebook-beacon.html
[16] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings 2019 Network and Distributed System Security Symposium (NDSS 2019)*. Internet Society. https://doi.org/10.14722/ndss.2019.23386
[17] Meta. 2019. About Facebook Pixel | Facebook Ads Help Center. https://www.facebook.com/business/help/742478679120153?id=1205376682832142
[18] Zahra Moti, Asuman Senol, Hamid Bostani, Frederik Zuiderveen Borgesius, Veelasha Moonsamy, Arunesh Mathur, and Gunes Acar. 2024. Targeted and Troublesome: Tracking and Advertising on Children's Websites . In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1517–1535. https://doi.org/10.1109/SP54263.2024.00118
[19] Browser Options. 2024. Browser Options. https://www.selenium.dev/documentation/webdriver/drivers/options/
[20] Arnold Roosendaal. 2010. Facebook Tracks and Traces Everyone: Like This! *SSRN Electronic Journal* (2010). https://doi.org/10.2139/ssrn.1717563
[21] Betsy Schiffman. 2007. Facebook CEO Apologizes, Lets Users Turn Off Beacon. https://www.wired.com/2007/12/facebook-ceo-apologizes-lets-users-turn-off-beacon/
[22] Selenium. [n. d.]. WebDriver. https://www.selenium.dev/documentation/webdriver/

A Other Pixel Configurations

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009