COALITION AGAINST AD FRAUD

Definitions of Mobile Fraud Schemes

adjust

Introduction

The aim of this document is to provide advertisers, supply-side networks, third-party vendors and any other industry participant with a rounded technical overview of the current state of mobile performance ad fraud. In it, we will walk readers through the practical side of fraud, and describe how such schemes work in technical detail.

We hope that you can then use this information to get a better understanding of ad fraud, and become better equipped to deal with the issue at large.

All active CAAF members at the time of release agree to the nomenclature, definitions and methodologies described in this document.

CAAF Member List

Company Name	Website	Member since
Aarki	https://www.aarki.com/	2017
AdAction Interactive	https://www.adaction.mobi/	2017
AdColony	https://www.adcolony.com/	2017
Applift	https://www.applift.com/	2017
Chartboost	https://www.chartboost.com/	2018
Curate Mobile	https://www.curatemobile.com/	2018
Dynalyst	https://www.dynalyst.io/	2017
Fyber	https://www.fyber.com/	2017
Headway Digital	https://www.headwaydigital.com/	2017
i-mobile	https://www.i-mobile.co.jp/	2017
InMobi	https://www.inmobi.com/	2017
ironSource	https://www.ironsrc.com/	2017
Jampp	https://jampp.com/	2017
Liftoff	https://www.liftoff.io/	2017
Madvertise	https://madvertise.com/en/	2018
Moloco	https://www.molocoads.com/	2018
Motive Interactive	https://www.motiveinteractive.com/	2018
nend	https://nend.net/	2017
Persona.ly	https://persona.ly/	2017
Remerge	https://www.remerge.io/	2017
Revmob	https://www.revmobmobileadnetwork.com/	2018
Vungle	https://vungle.com/	2017
YouAppi	https://www.youappi.com/	2017
Zucks	https://zucks.co.jp	2017

The Definition of Mobile Ad Fraud

Mobile ad fraud is when an individual or group attempt to defraud advertisers, publishers or supply partners by exploiting mobile advertising technology. The objective of fraudsters is to steal advertising budgets from advertisers.

Mobile ad fraud can take a number of different forms, from faked impressions, click spam, or faked installs. As the industry develops to fight current fraud techniques, the methods used by fraudsters change to become more effective. For example, fraudulent publishers seeking to benefit from false impressions may stuff adverts into a single pixel, or deliberately align an advert out of view to generate views or impressions that never took place. Such examples weren't thought of even a few years ago.

Differentiation between Compliance Fraud and Technical Fraud

At Adjust we differentiate between technical fraud and compliance fraud. Technical fraud aims to exploit the underlying technology of an attribution and analytics platform (AAP) in order to poach attribution from their legitimate source, or introduce a fake conversion to trigger an attribution although no real install happened. Meanwhile, compliance fraud is where the terms of the underlying Insertion Order (IO) are bent or breached by the supply-side partner (or their affiliates) in order to be commissioned for actions that are prohibited by the IO.

An AAP can do little to prevent instances of compliance fraud, simply because the IO that records all compliance rules is not known to us. However, we will still cover different compliance fraud schemes in this document to make sure they are known and fully understood.

Definitions of Compliance Fraud Schemes

Compliance Fraud describes all fraud schemes that use loopholes in the Insertion Order (IO) contract between advertiser and supply side partner. This includes (but is not limited to):

- False targeting (especially geo-based targeting)
- Mixing in undesired traffic sources (incentivized, adult, redirect/pop, etc.)
- Intentful over delivery
- Unauthorized rebrokering of offers

Usually, this type of fraud is mixed into a campaign's legitimate delivery, so as not to raise an advertiser's suspicions.

Let's look each of these schemes in more detail.

False Targeting

The act of willful mistargeting by the supply-side partner or their vicarious agent, in order to be awarded with a higher commission tied to the campaigns targeting rules. Includes (but is not limited to) country, demographic and audience targeting.

Unsolicited Traffic Types

Traffic driven to an advertiser's campaign, in which the advertiser is explicitly named as undesired for the fulfilling of the IO. Typical traffic source exclusions are incentivized traffic[1], adult traffic[2], or otherwise un-approved creatives[3].

Unsolicited Over Delivery

Wilful over delivery on the campaign limits on the part of the supply side, usually aimed at being paid while being over budget. In most cases this starts slowly but becomes bolder over time when advertisers show that they are willing to pay for over delivery - especially if the advertiser pays full price.

One example of this is when an advertiser places an order for 300 installs a day, and the supply partner delivers 330 each day. They might start over delivering slowly, and the advertiser might pay because the results are small and of good quality. Later, groups might get into a heated argument as there could be 10,000 installs over delivered, with the deliverer expecting to be paid.

Unsolicited Re-brokering

This is one of the greatest risks in the performance advertising space. Rebrokering between blind networks is problematic, as it removes control on who is delivering on a campaign and how it is done. Usually, the IO breaches mentioned above are a lot more pronounced on rebrokered campaigns, because the feeling of responsibility is diluted (as there is no direct partnership at risk).

---

1 Traffic wherein the delivery of ad media or a desired user action (e.g. an install of the advertised app or a certain action taken within the app) is incentivized by the supply-side partner paying out part of the conversion commission (CPI) to the end user/customer usually in the form of virtual goods (e.g. premium currency in a free to play game or points on a cashback or discount portal).

2 Traffic that originates from content that is aimed at adult/mature only consumption. Usually refers to pornographic content.

3 Un-approved creatives include "scareware" that mislead the user into converting under false pretenses

Definitions of Technical Fraud Schemes

Technical fraud schemes are defined by their efforts to try and manipulate AAPs into incorrect attribution. Technical fraud schemes can be separated into two sub-categories.

1. Install/conversion fraud - where conversion points that were never actually reached, like Fake Installs or fake revenue events, are pushed into the system to trigger an attribution for monetary gain.

2. Attribution fraud - where fraudsters try to manipulate attribution of legitimate conversion events to sources that have no hand in converting the customer to take the desired action, by forcing, faking or automatic customer engagements (impressions, views and clicks) without actual interaction by the customer.

Such schemes include Click Spam, Click Injection and Fake Installs. Let's look at these in more detail.

Click Spam

Also known as "Click Flooding", "Click Fraud" or "Fake Clicks", this term describes any fraud scheme that executes clicks on behalf of the device's user without the user's knowledge, consent or intent.

There are two types - high-frequency and low-frequency click spam.

High-Frequency Click Spam

Fraudsters with little reach (meaning a small active user base for an app or mobile web content) improve their chances to cash-in on random installs by spamming their users at high frequency, hoping to take the last click inside the attribution window. The very low conversion rate originates from up to hundreds of clicks a day for the same advertisement (tracker) on the same device (fingerprint/device ID).

Low-Frequency Click Spam

Fraudsters here have a large active user base for their app or mobile web content, and receive many unique visitors. Because of this, they can execute forced clicks less often, but still successfully create revenue on the random chance that users will convert for the apps the fraudsters are "running campaigns" on. Conversion rates for this kind of exploit are exceptionally low because a high amount of devices (fingerprint/device ID) are tagged per advertisement (tracker).

Two Examples

"Click Caching" or "pre-clicks"

In this instance of click spam, perpetrators would go ahead and let their app generate a click and cache the redirect link in order to be able to redirect to user - in case the user actually clicks an advertisement - to the pre-cached redirect URL. The problem for the advertiser (and the lucky happenstance for the traffic source) here is that MMPs will already have tracked a click for everytime this caching is executed, meaning that end users with the perpetrating app installed will have clicks generated in their name for dozens of ads that they will never have seen.

As a result, any actions that a user takes installing any of the apps that an advertisement and redirect were cached for will be attributed as if the user saw advertising and clicked it with the intent of finding out more about the advertised app.

Note: This method is sometimes justified as a user experience enhancement, with publishers claiming that redirects take too long when a click on ad media needs to be redirected through multiple networks to reach the App Store. But, actually, it is completely unnecessary to build this kind of workaround since Adjust (and the other major tracking providers) offer proper solutions to this, such as parallel click execution or server-to-server clicks.

Clicks on impression or view

In this example, any instance in which the publisher or network generates a click when the advertisement was rendered, or during or at the end of a video view without the user actually clicking. In many cases this is seen as a necessity to stay competitive with other supply side vendors, who are using the same tactic to stay ahead. Regardless of the situation in the market, the IAB/MRC measurement guidelines are clear about when a click is to be measured[4] and what makes it valid.

A click without intentful user interaction is not a valid click.

[4] https://www.iab.com/wp-content/uploads/2015/06/click-measurement-guidelines2009.pdf

Click Injection

Click injections are a more sophisticated approach to stealing attribution from organics (or paid channels, for that matter). They work by generating a click that did not originate from a user interacting with advertising with the intent of learning more about the advertised product. The defining feature of this approach is that a single click is enough to get the job done, as it will be injected after the user has already made the decision to download and try a new app.

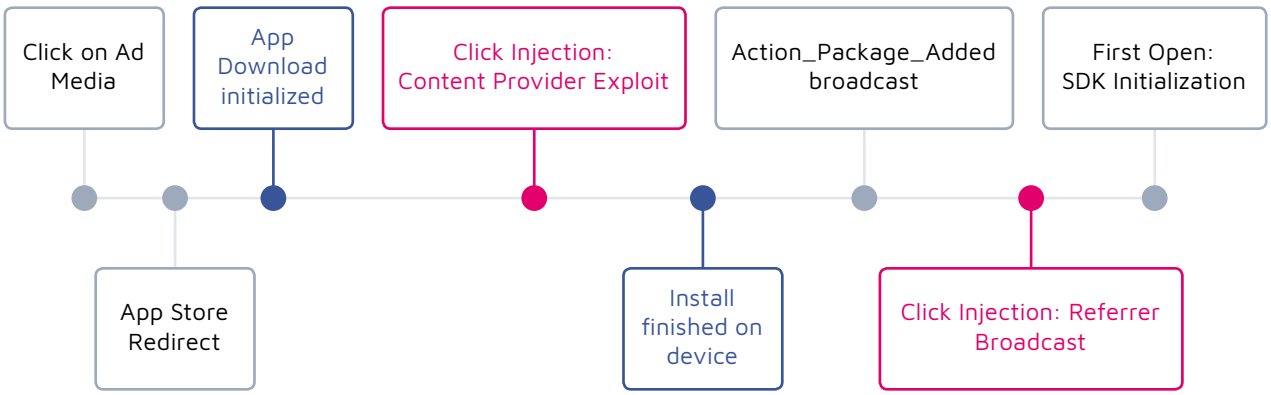
Package_Added Broadcast Exploit

The Android broadcast `ACTION_PACKAGE_ADDED` is used to receive a notification about a new app being installed on a device the fraudsters have their app installed on. From the broadcast, the perpetrator can determine which app has been installed, and look up if they have a matching offer link (click or impression).

Then, they fire it either directly from the respective device or server-to-server. The click will be processed and used for attribution. As a result, the perpetrators can steal organic install attributions as well as attributions to paid sources.

Content Provider Exploit

In this case, the perpetrators subscribe to a Google Play content provider (SQLite DB, part of the Android OS) in order to get notified about any app downloads being started, gaining access to all the data needed to inject a click for the right target app.



Example

A fraudster has an app ("CI-FRA-APP") in the Play Store that is supposed to be monetized for advertising. To boost revenues, they decide to use click injections to poach attribution for installs they had no part in convincing users to download.

A random user has "CI-FRA-APP" installed for its main purpose and is unaware of its secret monetization tactics.

The same user now organically installs a different app ("fun-game-app") which was introduced to her by a friend.

EITHER within a couple of milliseconds after the user clicked the download button, the content provider will notify the perpetrator's app of a new app being downloaded, allowing a click to be injected.

OR within a couple of milliseconds after the install on her device finishes, the broadcast `ACTION_PACKAGE_ADDED` will trigger on her device - and all apps on the device will be able to listen to it, including "CI-FRA-APP".

"CI-FRA-APP" will capture all the necessary device information and the bundle-name/ID of the newly downloaded or installed "fun-game-app", and will send these to their backend.

The fraudsters backend will find the offer link for "fun-game-app" in their database and create a valid click link that includes all the information needed for attribution (clickID, offer-code, device ID) and execute the link server-to-server (s2s) to their network partner or exchange. The redirect chain will be followed s2s until it reaches Adjust's servers, which should usually take less than a second.

OR

The fraudster has the correct link cached in their "CI_FRA_APP" or requests it from their backend to fire it directly from the device in a hidden web view. Again, the redirect chain would be followed properly, in less than a second, and the click information would reach Adjust's servers.

During simple attribution without any filters AAPs prefer the click with the shortest time to first open, and thus would attribute to the last click (rather than organics), and also to an injected click over a legitimate click, because it's closer to the first open.

If the user now opens the "fun-game-app" immediately after the install finished, when the icon appeared on her home screen, we would see a click-to-first-open-conversion time of only a few seconds.

Note

1. There are valid cases in which the click-to-first-open-time is in low single digits (seconds) but no click injections took place.

   For example, a user might replace or upgrade their device and re-download all their previously installed apps. Since the new device has a new advertising identifier, we would count all these re-downloads as new installs. If those re-downloads are opened without a relevant attribution action, such as a click or an impression, we would count them as new organic installs. However, if those apps are opened via a deep link, Google search, or a click on a banner that redirects to the App Store, which then displays 'open' instead of 'download', we would see a very short click-to-first-open-time. It is arguable that clients should not be paying for these attributions but it is clear these are not fraudulent in nature.

   Also as another example, with incent traffic a user might install an app from an offerwall in order to receive a reward but never open it until returning to the offerwall, seeing they did not receive their reward, and then clicking again (registering a new click) before atlast opening the app for the first time. A similar narrative can be written with video ad placements, which can inspire users with a previously downloaded app to open the app for the first time.

2. iOS does not have an equivalent to the content provider and dropped broadcasts as of the release of iOS 9. With iOS, in order for apps to communicate with one another, the active app needs to declare which other apps it wants to communicate with, and the list must both be conservative and make sense in order to pass inspection during the app review process. This makes click injections harder and riskier to set up, as the penalty for the fraudulent publishing company is exclusion from the Apple App Store.

3. It is possible that a monetization SDK could be the perpetrator of click injections without the app developer or publisher knowing their app is being used with malicious intent.

4. The quality of traffic coming from stolen organic installs will typically appear higher than that of paid UA. This dilemma often leads advertisers to invest more marketing resources into such channels, even though doing so is not in their own self-interest.

Fake Installs

This fraud scheme deals with any type of install where the sole purpose of that install is to trigger a commission (commonly CPI/CPA) for the fraudster. There is never a real user or a real device (as in belonging to a real user and used for a multitude of tasks) involved in these - actually, the devices are virtual or emulated.

This fraud scheme originated from similar practices on desktop web/affiliate marketing, where conversion points were faked to score commissions. Back then - and even now - fake leads have been playing a big part in the fraud composition and even fake sales have been created.

In the mobile app environment fake installs make up the majority of fake conversion fraud that is currently stealing budgets, but there have also been cases of fake in-app purchases (IAP) that lead to (cost per action) (CPA) payouts.

Fake Installs from Emulation

Fraudsters use commonly available device emulation software in virtualized environments (on server hardware) to fake installs in an effort to claim advertising revenue to great effect, programming scripts that make the emulator create a new random device with a fresh Device ID.

On that device, they can then create a user, and have that user engage with advertisements. The emulated device will download the target app from an app store (or from local storage to cut down on traffic cost), thereby triggering an install. Finally, the emulated device will open the installed app to trigger an install event, which is then transmitted to the attribution provider.

Sophisticated fraudsters might even go as far as storing the session for later use to create third or seventh-day retention by opening another session at the desired time.

Usually, the perpetrators will route all traffic from the data center through different anonymizing services such as VPNs, public or private proxies or the TOR network or any variation thereof.

Fake Installs from Device/Install Farms

This scheme is similar to the above 'Fake installs from Emulation'. In this variation, the fraudsters actually have physical devices present at their place of business. Instead of scripts providing the action input for emulated devices in a device farm, the action might be human interaction (with fraudulent intent) or partial scripting of the devices through a controlling computer. The devices then get regularly reset to different possible extents, e.g. full factory reset, Advertising ID reset or reapplied custom ROM, in order to trick AAPs to register multiple installs on the same device as new installs.

Again, the traffic from these device/install farms will be routed through different anonymizing services such as VPNs, public or private proxies or the TOR network or any variation thereof.

Fake Installs from SDK Spoofing

SDK spoofing (or replay attacks) is a form of mobile performance fraud that consumes an advertiser's budget by generating legitimate-looking installs without any real installs occurring.

This type of fraud evolved very quickly and dramatically during the course of 2017. This is because SDK spoofing has become harder to spot than fake installs generated in emulation or install farms, as the devices used in this scheme are real - and so are normally active and spread out.

This type works by fraudsters using a real device without the device's user actually installing an app. The perpetrators' main approach was to break open the SSL encryption between the communication of a tracking SDK and its backend servers, typically done by performing a 'man-in-the-middle attack' (MITM attack). Now, the most popular approach is to use a proxy software (e.g. Charles Proxy).

After completing the MITM attack, fraudsters would then generate a series of test installs for an app they want to defraud. Since they can read the URLs in clear text format for all the server-side connections, they can learn which URL calls represent specific actions within the app, such as first open, repeated opens, and even different in-app events like purchases, levels up or anything else being tracked. They also research which parts of these URLs are static and which are dynamic, keeping the static parts (things like shared secrets, event tokens, etc) and experimenting with the dynamic parts, which include things like advertising identifiers or other data specific to the device and the particular circumstances.

Now, thanks to callbacks and near real-time communication detailing the success of installs and events, the perpetrators can test their setup by simply creating a click and a matching install session. If the install doesn't go through, then there is a mistake in their URL logic. If it is successfully tracked, they've cracked the logic.

Once an install is successfully tracked, the fraudsters will have figured out a URL setup that allows them to create installs from thin air.

Methods continue to change - we see fraudulent device data matching data from real-device traffic, consistent over a multitude of device-based parameters (and, later, all device-based) parameters.

Now, not everything is fake. Fraudsters started to collect real device data. They did this by using their own apps or by leveraging any app they have control over. The intent of their data collection is, of course, malicious, but that does not mean that the app being exploited for data is purely malicious or could even be found out as malicious. The perpetrator's app might have a very real purpose, or it might be someone else's legitimate app. The perpetrators simply have access to it because their SDK is integrated within it. Regardless of the specific circumstances, the fraudsters have access to an app that is being used by a large number of users.

Having a source (or even multiple sources) that generates real device data makes the fraudsters' task simpler. They no longer need to randomize or curate large amounts of data, because they have access to the real thing.

This evolution went hand in hand with a second impactful step in how much more sophisticated SDK spoofing became. The URLs no longer called from data centers, or tunneled through VPNs. Instead, they were proxied directly through the app the perpetrator had access to on an unsuspecting user's device.

This means a fraudster's server runs a script that automatically creates a URL that will trigger us (or any attribution company) to track an install or event. Instead of sending this URL directly to our servers (or through an anonymizing network as they used to) the fraudsters now send it to the app (the one the perpetrators have access to) on a user's device. This app then executes the URL on the user's device.

We hope with this short document you've learned more about the types of fraud that are currently perpetrated within the market. If you'd like to talk to us or have any questions, please contact us directly.