Hackerone Bug Bounty Report: Modification of Assumed Immutable Data (M.A.I.D) on the Hinge Dating Application

Abusing default settings in the Cloudinary Image Transformation API

Tyler Butler

Figure 1: Bug Bounty Details

Platform	HackerOne
Client	Hinge
Title	Profile Photo URL Manipulation Enables Modification of Assumed-Immutable Data
Severity	Low (3.7)
Weakness	Modification of Assumed-Immutable Data (MAID)
Bounty	$250

Abstract

Hinge is dating application for android and iOS devices launched in 2013. Like its competitors Tinder and Bumble, it enables users to search through a database of other users and match with potential dating partners. Offering features to create unique profiles, integrate with existing social platforms, and chat with other users, it uses a mixture of proprietary code and third-party services. This report outlines a low risk misconfiguration disclosed to Hinge through Hackerone in March of 2020 by Tyler Butler and triaged in June 2020. Hackerone is a bug bounty platform that connects freelance security researchers with clients to enable public and private security vulnerability disclosure[1].

The report outlines a potential vulnerability by exploiting improper media access controls in a third-party media storage provider used by Hinge. By abusing the default configuration of the *Cloudinary Image Transformation API*, original user images could be accessed in their original and unedited state. This posed a minimal risk to users who intend to use the crop feature to remove undesirable and potentially sensitive aspects of profile photos. While is unlikely that leaving the application in its disclosed state would result in widespread sensitive information leak, Hinge decided to triage the application to mitigate the risks involved. To exploit the flaw on a large scale, an attacker would need a highly customized script to automate and identify vulnerable users.

Scope

This report is constrained only to the Hinge iOS application (*app-id 595287172*), the *hinge.app.link* subdomain, and the *hinge-res.cloudinary* subdomain. While the security settings exploited in the Hackerone report applies to the Android app as well, it will not be discussed.

Testing Approach

The general approach to the security analysis conducted in this report involved two main steps. First, Burpsuite was used to intercept network traffic sent between a sample iPhone and the Hinge HTTP and HTTPS API endpoints. Burpsuite is an application security testing software produced by Portswigger. Intercepted traffic was dissected by endpoint, and manually analyzed. Second, the application was used on an iPhone as a normal user. Features which exposed endpoints were tested and all external links were collected and analyzed. To protect Hinge user's personal information, exploit POC documentation included in this report uses sample Cloudinary assets not associated with Hinge.

Application Design

The Hinge platform consists of native applications for both android and iOS devices. In addition to integration with the Facebook-owned social media platform Instagram, there are also several third-party Software as a Service (SaaS) providers used to supplement custom development. According to data analyzed from BurpSuite packet captures Hinge uses SendBird, a chat as a service platform; Braze, a CRM and marketing platform; and Branch, a mobile measurement and deep-link solution platform.

Central to application is the ability for users to create custom profiles using a mixture of user-uploaded images, captions, questionnaire answers, and link to their Instagram feed. Storage of user images is done through Software as a Service (SaaS) provider, *Cloudinary*. Users are required to upload six images when first creating an account, and the first image displayed on the profile is used as the user's profile picture.

Third Party Services

Cloudinary

The Cloudinary platform offers image, video, asset, and media management solutions for developers and e-commerce[2]. It is used in photo-heavy applications because of its easy to use API. Best summed up on the Cloudinary website, the API enables users to,

"apply artistic effects to an image or to simply scale it. With simple system of chained transformations, you can crop, scale, transcode, filter, and optimize your original high-resolution images on the fly. Tailor transformations

[1] The Most Trusted Hacker-Powered Security Platform. (n.d.). Retrieved June 17, 2020, from https://www.hackerone.com/

[2] Image and Video Management Solutions | Cloudinary. (n.d.). Retrieved June 17, 2020, from https://cloudinary.com/solutions

based on conditional parameters or the viewing context to deliver the most appropriate version to users"[3].

The ability to dynamically display assets based on conditions means that assets can be displayed optimally regardless of their environment. This can commonly be seen in the use of thumbnail images. When the image needs to be displayed as a thumbnail or icon, it will dynamically be optimized to retain its essential aspects despite being reduced in size. Cloudinary clients include Buzzfeed, CNN, and Uber among others.

Cloudinary Image Transformation API

Transform Parameters

Each asset in the Cloudinary media library is given a unique public ID and is accessible through their website at the cloudinary.com/image/upload directory. Cloudinary uses dynamic URL's to apply effects to images on its platform. To apply effects, transformation parameters can be appended to the address of the asset. In total, 33 different transformation parameters can be used for a variety of purposes such as changing height and width, effects, backgrounds, and custom functions among others.[4] An example of a transformed image on Hinge looks like the following,

"https://hinge-res.cloudinary.com/image/upload/x_0.18,y_0.21,w_0.36,h_0.27,c_crop/w_1055,q_auto/f_webp/[profile number]/[unique profile id].jpg"

When users make a change to an image hosted on Cloudinary, new transformation requests are added on the fly. Original image assets themselves are not changed, however, transformation parameters applied to the request URL change what is displayed when requested. This distinction means that both the original asset as it was uploaded to the platform as well as the new requested URL with transformations are able to be requested and exist available to the public. This is an intended function of Cloudinary and a default setting, however, this can be changed through the enforcement of *Media Access Controls* such as Strict Transformation and Signed Delivery URLS[5].

Media Access Controls

One media access control method to mitigate the risk of user requested transformations is *Strict Transformations*. Through this feature, application owners can explicitly define which, if any, transformations can be requested by users. Another access control method is the *Signed Delivery URL* option. With this option, images are validated using a URL signature which is a base64 encoding of a SHA1 digest made from the application owners image ID and transformation string chained with the API secret. Only images which pass authentication can be viewed, meaning access to images can be controlled for private/authenticated viewing or manipulation.

Vulnerability Discovery

The risk described in this report and submitted to Hinge relies on the lack of use of media access controls on user images. In fact, by default, no access controls are implemented on the Cloudinary platform. In effect, this means that any asset uploaded to the platform should be accessible in its original state regardless of what users cropped out in the app. To exploit these settings, the URL address of user profile pictures needed to be located. This was achieved using the iOS app "share profile" option. This feature enables users to share profiles with others. The feature generates a standard message, "I recommended [Username] for you on Hinge! Log into the app to view their profile", as well as a URL link to the user's profile with the format,

"https://app.hinge.co/[User profile ID]" .

When shared through SMS messaging and opened on mobile devices with Hinge installed, the link triggers the application to open. If Hinge is not installed, the user will be re-directed to the app store. Users are automatically guided in this way from the use of iOS deep links, a feature implement for iOS in 2009[6]. Figure 2 shows the hinge.co subdomain which hosts the Universal Link json file necessary for such

Figure 2: Hinge Universal Link JSON File


```
{
    "applinks": {
        "apps": [],
        "details": [
            {
                "appID": "ZJ7R546CUS.co.hinge.mobile.ios",
                "paths": [ "*"]
            },
            {
                "appID": "9UCJ3BPV5V.co.hinge.ios.Hinge",
                "paths": [ "*" ]
            }
        ]
    }
}
```


Figure 2: Configuration file located at : https://hinge.co/.well-known/apple-app-site-association

configurations. User profiles are also able to be viewed with a browser. Browser-based Hinge profiles feature only the username, a comment, and a profile photo. Browser access to the user profile photo allowed for easy access to the custom Hinge Cloudinary domain, *hinge-res.cloudinary.com*.

[3] Image Manipulation | Cloudinary Features. (n.d.). Retrieved June 17, 2020, from https://cloudinary.com/features/image_manipulation

[4] Image transformations. (n.d.). Retrieved June 17, 2020, from https://cloudinary.com/documentation/image_transformations

[5] Media access control. (n.d.). Retrieved June 17, 2020, from https://cloudinary.com/documentation/control_access_to_media

[6] Universal Links - Apple Developer. (n.d.). Retrieved June 17, 2020, from https://developer.apple.com/iOS/universal-links/

Figure 3: Overlay Class Containing Profile Image Asset

Figure 3: Snippet of profile URL

As seen in Figure 3, profile pictures are loaded into the browser-based profile page through the hinge-res subdomain of Cloudinary. The URL extension, including the various tags denoted with backspaces, are a part of the *Cloudinary Image Transformation API*.

Exploiting the Transform API

With no media access controls in place, exploitation of the transform API is fairly straight forward. By removing the transformation parameters to the GET request URL, the original asset can be seen as it was uploaded by the user.

Figure 4: Changing GET Request Parameters

Requesting Transformed Asset

"https://hinge-res.cloudinary.com/image/upload/x_0.18,y_0.21,w_0.36,h_0.27,c_crop/w_1055,q_auto/f_webp/[profile number]/[unique profile id].jpg"

Requesting Original Asset

"https://hinge-res.cloudinary.com/image/upload/q_auto/f_webp/[profile number]/[unique profile id].jpg"

Figure 4: Transformation parameters are removed from URL address

Risks

The risks involved in leaving original user assets available publicly are low. Hackerone classified this finding as a 3.7 on Hackerone's severity scale. Still, some of the findings using this method to view original images were interesting. Some user's uploaded screenshots of their personal mobile devices, and only after upload edited the image to zoom in on a face or body. The full screen shot exposed other information about those users. In one example, a screenshot of the users Instagram profile revealed their name and Instagram handle. Other users uploaded images in various stages of undress and used the crop feature to crop out features other than their face. Commonly, many users used the crop feature to crop themselves out of images with others, including large group photo's or photos with significant others.

While no seriously damaging information or images was found through this method, it does not take much extrapolation to see how in the wrong hands, a determined attacker could scrape these images and find a subset that the user would not want available online.

Proof of Concept

To demonstrate accessing original user assets, and to protect the information and identify of Hinge users, a limited proof of concept has been created using a free account on the Cloudinary platform. After creating a free account, a public domain image was uploaded to the account's media library.

Next, using the Transformations dashboard, a new transformation was created which adjusts image width to 461 pixels, moves the x axis to position 190 and the y axis to position 0. The final transformation request looks like the following "c_crop,w_461,x_190,y_0". With *Strict Transformations* not enabled, the transformation was applied to the sample image . The new transformed image was browsed at the footnoted address and is shown in Figure 5.[7]

Figure 5: Transformed POC Image

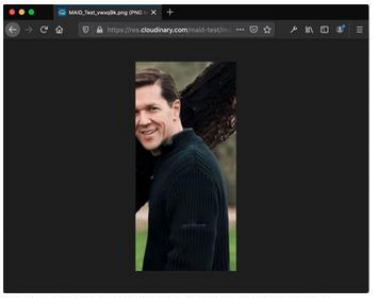

Figure 5: The image as it would be seen through a transformed state

To demonstrate the flaw with default Cloudinary settings, the transformation parameter was removed from the requested address. The original image was browsed using the footnoted address and is shown in Figure 6.[8]

Figure 6: Original POC Image

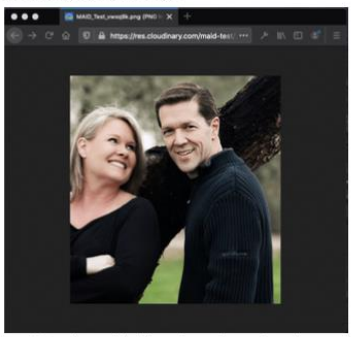

Figure 6: Revealing an original image by removing transformation parameters

[7] MAID Test [Digital image]. (n.d.). Retrieved from https://res.cloudinary.com/maid-test/image/upload/c_crop,w_461,x_190,y_0/v1587170581/MAID_Test_vwxq9k.png

[8] MAID Test [Digital image]. (n.d.). Retrieved from https://res.cloudinary.com/maid-test/image/upload/v1587170581/MAID_Test_vwxq9k.png