

Instantly share code, notes, and snippets.



hackermondev / research.md

Last active 30 minutes ago

Code

Revisions 3

Stars 553

Forks 35

Embed ▾

<script>



Download ZIP

Unique 0-click deanonymization attack targeting Signal, Discord and hundreds of platform

research.md

hi, i'm daniel. i'm a 15-year-old high school junior. in my free time, i [hack billion dollar companies](#) and build cool stuff.

3 months ago, I discovered a unique 0-click deanonymization attack that allows an attacker to grab the location of any target within a 250 mile radius. With a vulnerable app installed on a target's phone (or as a background application on their laptop), an attacker can send a malicious payload and deanonymize you within seconds--and you wouldn't even know.

I'm publishing this writeup and research as a warning, especially for journalists, activists, and hackers, about this type of undetectable attack. Hundreds of applications are vulnerable, including some of the most popular apps in the world: Signal, Discord, Twitter/X, and others. Here's how it works:

## Cloudflare

By the numbers, Cloudflare is easily the most popular CDN on the market. It beats out competitors such as Sucuri, Amazon CloudFront, Akamai, and Fastly. In 2019, a major Cloudflare outage knocked most of the internet offline for over 30 minutes.

One of Cloudflare's most used feature is Caching. Cloudflare's Cache stores copies of frequently accessed content (such as images, videos, or webpages) in its datacenters, reducing server load and improving website performance (<https://developers.cloudflare.com/cache/>).

When your device sends a request for a resource that can be cached, Cloudflare retrieves the resource from its local datacenter storage, if available. Otherwise, it fetches the resource from the origin server, caches it locally, and then returns it. By default, [some file extensions](#) are automatically cached but site operators can also configure new cache rules.

Cloudflare has a vast global presence, with hundreds of datacenters in 330 cities across 120+ countries—an estimated 273% more datacenters than Google. In the U.S. East region, for example, the nearest datacenter to me is less than 100 miles. If you live in a developed country, there's a good chance the nearest datacenter to you is less than 200 miles from you.

A few months ago, I had a lightbulb moment: if Cloudflare stores cached data so close to users, could this be exploited for deanonymization attacks on sites we don't control?

You see, Cloudflare returns information about a request's cache status in the HTTP response.

|                  |                         |
|------------------|-------------------------|
| Cache-Control:   | public, max-age=2592000 |
| Cf-Cache-Status: | HIT                     |
| Cf-Ray:          | 9049a14f2f86c985-IAD    |

cf-cache-status can be HIT / MISS and cf-ray includes the airport code for the closest airport to the datacenter that handles the request (in my case, IAD).

If we can get a user's device to load a resource on a Cloudflare-backed site, causing it to be cached in their local datacenter, we can then enumerate all Cloudflare datacenters to identify which one cached the resource. This would provide an incredibly precise estimate of the user's location.

## Cloudflare Teleport

---

There was a one major hurdle I had to get through before I tested this theory.

You can't simply send HTTP requests to individual Cloudflare datacenters. For "security purposes" (presumably DDoS protection), all Cloudflare IP ranges are strictly anycast. All TCP connections opened to their network are always handled by the nearest available datacenter to you, there's no way you can ask a datacenter in Canada to handle your request if you live in the US.

However, after some research, I found a forum post (<https://community.cloudflare.com/t/how-to-run-workers-on-specific-datacenter-colos/385851>) from a community member showing me exactly how.

The author shared a bug he found to send requests to specific Cloudflare datacenters with Cloudflare Workers.

I'm still not 100% sure of the specifics of this bug, but using an IP range used internally by Cloudflare WARP (Cloudflare's VPN client), we could ask certain datacenters to handle HTTP requests. Normally, this IP range blocked inbound connections from external IP addresses but requests sent from Workers could bypass this since the connection would originate from inside Cloudflare's network.

I spent a few minutes reading their post and I quickly spun up a tool for this: Cloudflare Teleport (<https://github.com/hackermandev/cf-teleport>). Certain IP ranges corresponded to different datacenters (<https://github.com/hackermandev/cf-teleport/blob/main/scripts/data/colos.json>).

Cloudflare Teleport is a proxy powered by Cloudflare Workers that redirects HTTP requests to specific datacenters. For example, <https://cfteleport.xyz/?proxy=https://cloudflare.com/cdn-cgi/trace&colo=SEA> would proxy a HTTP GET request to <https://cloudflare.com/cdn-cgi/trace> specifically to a Seattle (SEA) datacenter.

Cloudflare would end up completely patching this bug a few months later, making this tool obsolete, but more on that later. For a majority of my initial, I used this tool.

## First "Deanonymization" Attack

As soon as the Cloudflare Teleport tool was complete, I was able to confirm my theory. I coded a simple CLI program that would send an HTTP GET request to a specified URL and list all datacenters that had the resource cache and its age.

For my first test, I used Namecheap's favicon (<https://www.namecheap.com/favicon.ico>). This resource has Cloudflare Caching enabled, it's just a simple static image of their logo. (This was the quickest site I could find that didn't have rigorous bot protection):

```
hackerman@hackerman cfteleport-cli % ./dist/cli.js single https://www.namecheap.com/favicon.ico
HIT EWR (Newark, NJ, United States) 200 https://www.namecheap.com/favicon.ico, age: 4 minutes 235 seconds, latency: 1925ms
HIT IAD (Ashburn, VA, United States) 200 https://www.namecheap.com/favicon.ico, age: 4 minutes 235 seconds, latency: 1964ms
HIT BOS (Boston, MA, United States) 200 https://www.namecheap.com/favicon.ico, age: 4 minutes 233 seconds, latency: 1559ms
HIT YYZ (Toronto, ON, Canada) 200 https://www.namecheap.com/favicon.ico, age: 4 minutes 226 seconds, latency: 2590ms
HIT MIA (Miami, FL, United States) 200 https://www.namecheap.com/favicon.ico, age: 3 minutes 174 seconds, latency: 1923ms
HIT MAN (Manchester, United Kingdom) 200 https://www.namecheap.com/favicon.ico, age: 3 minutes 157 seconds, latency: 2263ms
HIT SEA (Seattle, WA, United States) 200 https://www.namecheap.com/favicon.ico, age: 3 minutes 157 seconds, latency: 2018ms
HIT LAX (Los Angeles, CA, United States) 200 https://www.namecheap.com/favicon.ico, age: 2 minutes 141 seconds, latency: 1953ms
HIT SJC (San Jose, CA, United States) 200 https://www.namecheap.com/favicon.ico, age: 2 minutes 106 seconds, latency: 1902ms
HIT LHR (London, United Kingdom) 200 https://www.namecheap.com/favicon.ico, age: 1 minute 68 seconds, latency: 2089ms
HIT ORD (Chicago, IL, United States) 200 https://www.namecheap.com/favicon.ico, age: 47 seconds 47 seconds, latency: 2026ms
HIT ATL (Atlanta, GA, United States) 200 https://www.namecheap.com/favicon.ico, age: 34 seconds 34 seconds, latency: 1547ms
HIT NRT (Tokyo, Japan) 200 https://www.namecheap.com/favicon.ico, age: 27 seconds 27 seconds, latency: 2113ms
```

Boom, it worked. Namecheap had configured their cache age extremely low (5 minutes) but I was able to see every datacenter that had cached the their site's favicon in the last 5 minutes. Since everytime you load their site, your browser automatically downloads this favicon, this means a user from each one of this locations has visited the Namecheap.com site within the 5 minutes with the last visit from Tokyo, Japan.

This was just meant to be a simple test and there's almost no impact here, but with this I confirmed my theory. This proved the concept of using Cloudflare caching for deanonymization attacks.

## Real-World Application: Signal

Signal, an open-source encrypted messaging service, is widely used by journalists and activists for its privacy features. Internally, the app utilizes two CDNs for serving content: <cdn.signal.org> (powered by CloudFront) for profile avatars and <cdn2.signal.org> (powered by Cloudflare) for message attachments.

# 1-Click Attack

When a user sends an attachment (e.g., an image) on Signal, it is uploaded to cdn2.signal.org. Once the recipient opens the conversation, their device automatically downloads the attachment. Since Cloudflare caching is enabled for these URLs, an attacker can use the cache geolocation method to pinpoint the recipient's location.

The `https://cdn2.signal.org/attachments/*` path is configured to cache responses with Cloudflare. This means once a user's device automatically downloads an attachment, it's possible for an attacker to run a cache geolocation attack to find out which local datacenter they're near--similar to how law enforcement track mobile devices through cell phone towers.

To test this, I quickly patched the Signal desktop app to remove SSL pinning and configured Burp to intercept and view HTTP requests/responses sent through the app.

```
[hackermon@hackermon MacOS % HTTP_PROXY=http://localhost:8080 HTTPS_PROXY=http://localhost:8080 ./Signal]
```

## Reproduction Steps

### 1. Block HTTP GET requests to `cdn2.signal.org/attachments/*` in from the Signal using Burp.

This ensures that the app doesn't download attachments uploaded from the our side (the attacker) since that would cache them to our local datacenter and pollute the results. The best way I found to do this with Burp is to configure intercept rules for attachments, then leave request intercept on and deny all requests.

#### Request interception rules

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

Intercept requests based on the following rules:

| Add | Enabled                             | Operator | Match type     | Relationship        | Condition  |
|-----|-------------------------------------|----------|----------------|---------------------|--|
|     | <input type="checkbox"/>            |          | File extension | Does not match      | (^gif\$ ^jpg\$ ^png\$ ^css\$ ^js\$ ^ico\$ ^s...) |
|     | <input type="checkbox"/>            | Or       | Request        | Contains parameters |  |
|     | <input type="checkbox"/>            | Or       | HTTP method    | Does not match      | (get post)                                       |
|     | <input type="checkbox"/>            | And      | URL            | Is in target scope  |  |
|     | <input type="checkbox"/>            | And      | Domain name    | Matches             | cdn2.signal.org                                  |
|     | <input checked="" type="checkbox"/> | And      | URL            | Matches             | https://cdn2.signal.org/attachments/*            |
|     | <input checked="" type="checkbox"/> | And      | HTTP method    | Matches             | GET  |

### 2. Send an attachment (image) to a target.

This should work with any attachment but images are automatically downloaded when the user opens the conversation so they work best. I used a simple 1x1.png image for this test. The upload request is sent to Signal's CDN and you can see the attachment url in Burp once we send the attachment in the conversation and Signal uploads it (ex.

<https://cdn2.signal.org/attachments/UjLld11tvaL16M8mrd86>.

The screenshot shows a Signal conversation window on the left and the Burp Suite interface on the right. In the Signal window, a message from 'dan' says 'Missed video call - 4:00 PM' with a 'Call back' button. Below the message, there's a 'Today' section and a 'Message' button. In the Burp Suite 'Logger' tab, two requests are listed:

| #  | Time                 | Tool  | Method | Host            | Path                     | Query                  | Param count | Status code |
|----|----------------------|-------|--------|-----------------|--------------------------|------------------------|-------------|-------------|
| 51 | 08:27:19 20 Jan 2025 | Proxy | POST   | cdn2.signal.org | /attachments/UjLld11t... | X-Goog-Algorithm=GO... | 6           | 201         |
| 52 | 08:27:20 20 Jan 2025 | Proxy | PUT    | cdn2.signal.org | /attachments/UjLld11t... | X-Goog-Algorithm=GO... | 74          | 200         |

The 'Request' pane shows the PUT request details, and the 'Response' pane shows the HTTP/2 response with status 200 OK. The 'Inspector' pane on the right displays various request and response attributes.

### 3. Attack

After the targets opens the conversation (verify this with read receipts), their device should download the attachment which in turn causes Cloudflare to cache the file in a local datacenter.

I ran this attack on myself, used the CLI tool I mentioned earlier with the attachment url and found local datacenters that had cached the attachment.

```
[hackermon@hackermon cfteleport-cli % ./dist/cli.js single "https://cdn2.signal.org/attachments/Kqikr90jwCCZcSnjC7X5"
HIT EWR (Newark, NJ, United States) 200 https://cdn2.signal.org/attachments/Kqikr90jwCCZcSnjC7X5, age: 5 minutes 272 seconds, latency: 1015ms
```

In my case, I'm in New York and one of the closest datacenters to me is Newark, NJ (EWR) which is about 150 miles from my actual coordinates.

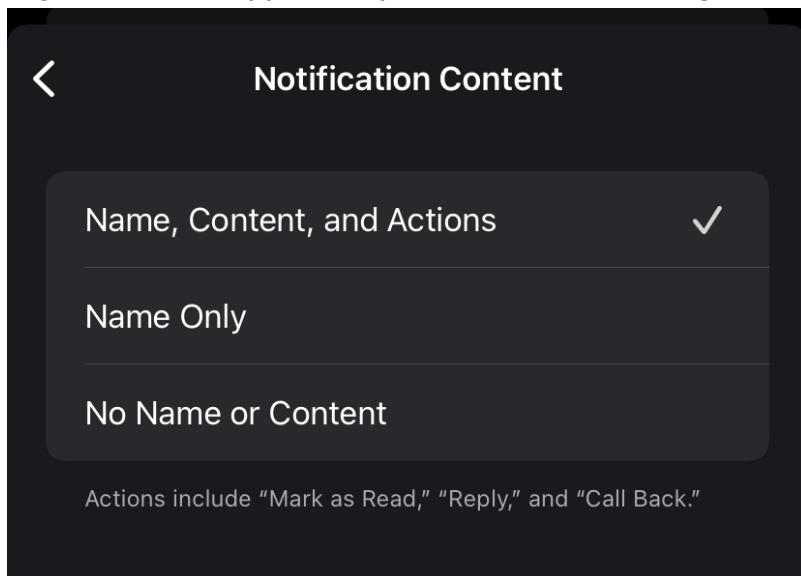
With an innocent-looking attachment, an attacker can deanonymize users and find their location within an approximate radius.

## 0-click

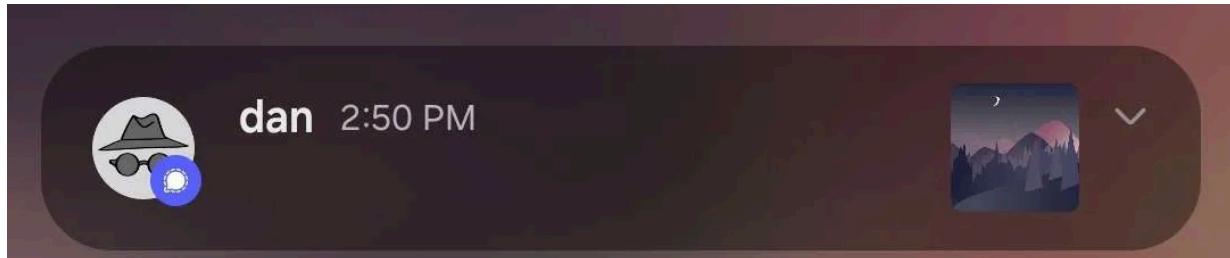
Here's where things get interesting. Although the 1-click method works, it requires the user to open the Signal conversation. Is it possible to run this attack without a single user interaction? Enter push notifications.

## Push Notifications

Signal's mobile app has 3 push notification settings.



Push Notifications are triggered if the user receives a message while not actively on the Signal app. By default, the mobile app includes the author, and message when it sends a push notification to your device.



The image shown on the right of the notification is the attachment sent with the message from Signal.

If the target has push notifications enabled (which it is by default), they don't even have to open the Signal conversation for their device to download the attachment. Once the push notification is sent to their device, it automatically downloads the image from Signal's CDN triggering the local datacenter to cache the response.

An attacker can run this deanonymization attack any time and grab a user's current location without a single interaction.

Signal, like Telegram, is used by journalists, activists, whistleblowers from all over the world. The potential for this attack is massive. This attack can be used to track Signal accounts, correlate identities, find employees meeting with journalists and much more.

## Real-World Application: Discord

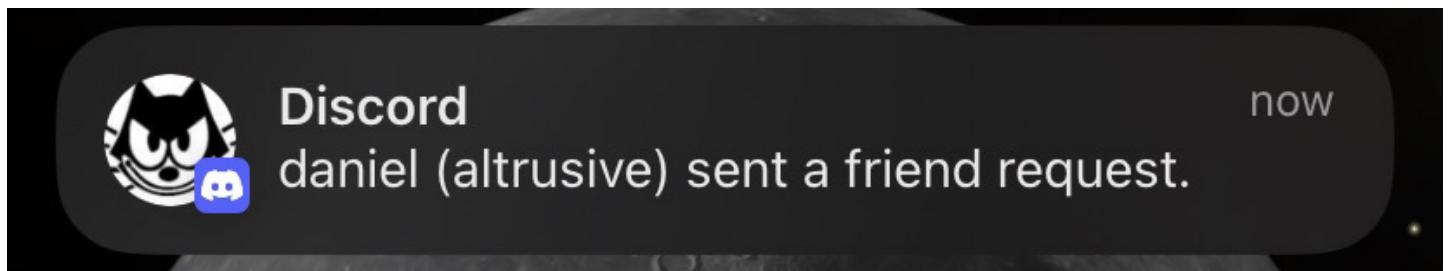
During my research, another app I found vulnerable to this type of attack is Discord. Discord is a free app that allows users to communicate with each other through text, voice, and video. Although the app is targeted towards gamers, Discord has been in the news recently this past year for facilitating government leaks and Discord hosts a significant portion of cybercrime on the internet.

The 1-click aspect is very simple and fairly similar to Signal, I would say the impact is even wider with Discord. Discord allows users with a Nitro subscription (their \$9.99/mo premium service) to use custom emojis in a variety of places: Messages, User Presence, Channels, etc. These custom emojis are loaded from Discord's CDN and are configured to be cached on Cloudflare. An attacker can use the same deanonymization attack with Signal to deanonymize users.

So, instead of sending an attachment in a Discord channel, an attacker can display a custom emoji in their user status and simply wait for the target to open their profile to run a deanonymization attack.

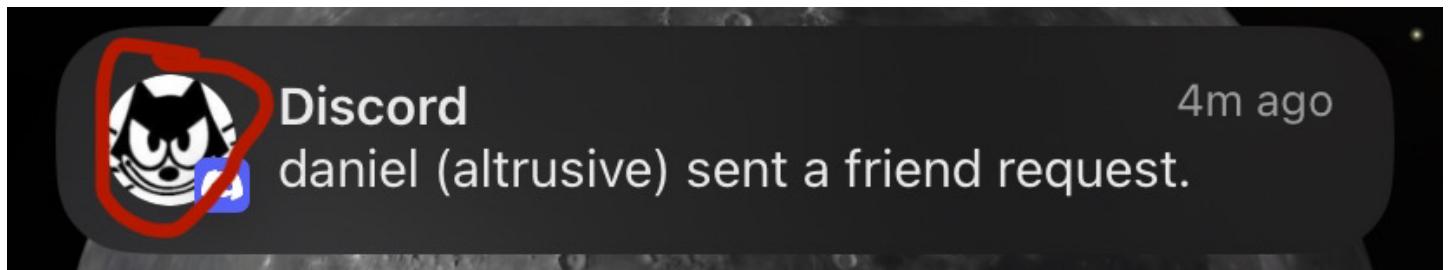
I've disclosed the [entire HackerOne report](#) I sent to Discord which has specific details, but I want to focus on the 0-click aspect here.

In Discord, mobile push notifications are sent for a variety of events (not just for messages received like Signal). For example, sending a friend request to a Discord user triggers a push notification on the user's mobile device.



Interestingly, even if the user is actively on Discord, friend request notifications are always sent to the user's mobile device.

How would a deanonymization attack be used with a friend request notification? Well, take a look at the notification.



When you receive a friend request and Discord sends the notification to your device, it includes the user's avatar URL to be shown with the notification. Your phone downloads the avatar URL (without any user interaction) and displays it alongside the notification.

Discord has Cloudflare caching configured on the CDN path for avatar URLs, which means we can simply do the same cache location attack mentioned earlier.

In Discord, the avatar URL format used in push notifications is:

[https://cdn.discordapp.com/avatars/{user\\_id}/{avatar\\_hash}](https://cdn.discordapp.com/avatars/{user_id}/{avatar_hash}) . The avatar URL format used in the website to display user avatars is slightly different (it always contains an image extension) ([https://cdn.discordapp.com/avatars/{user\\_id}/{avatar\\_hash}.png](https://cdn.discordapp.com/avatars/{user_id}/{avatar_hash}.png) ).

Both URLs lead to the same image but since images displayed in the app have a different path, they're cached separately. This ensures our results are not polluted and allows us to ensure we are finding the datacenter of a device that loaded the avatar through a push notification and not just the profile on the Discord app.

Just like that, we have the steps for a 0-click version of this attack for Discord:

1. Change your user avatar. This randomizes your avatar hash and ensures your avatar URL has not been loaded by anyone yet, increasing the accuracy of the attack.
2. Send a friend request to the target. Although there's a variety of ways to trigger push notifications with Discord, I choose friend requests because they are always sent regardless of whether the user is active on Discord. They also don't require any mutual server with the target, meaning you can practically do this with anyone on Discord.
3. Use Cloudflare Teleport tool on the user avatar and find all local datacenters that have cached the avatar

## GeoGuesser

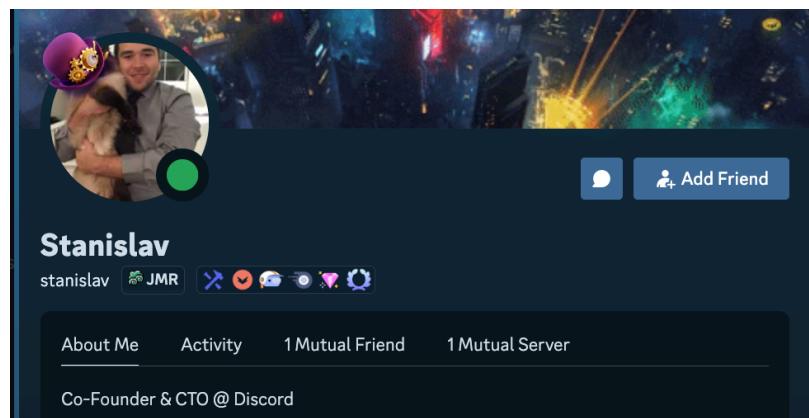
---

I'm very familiar with Discord's API and I realized I could automate every step in Discord's 0-click attack, and so I did.

Introducing GeoGuesser. This is a private Discord bot with a single command that takes a username, runs an attack with the steps mentioned earlier and returns the result entirely through Discord.

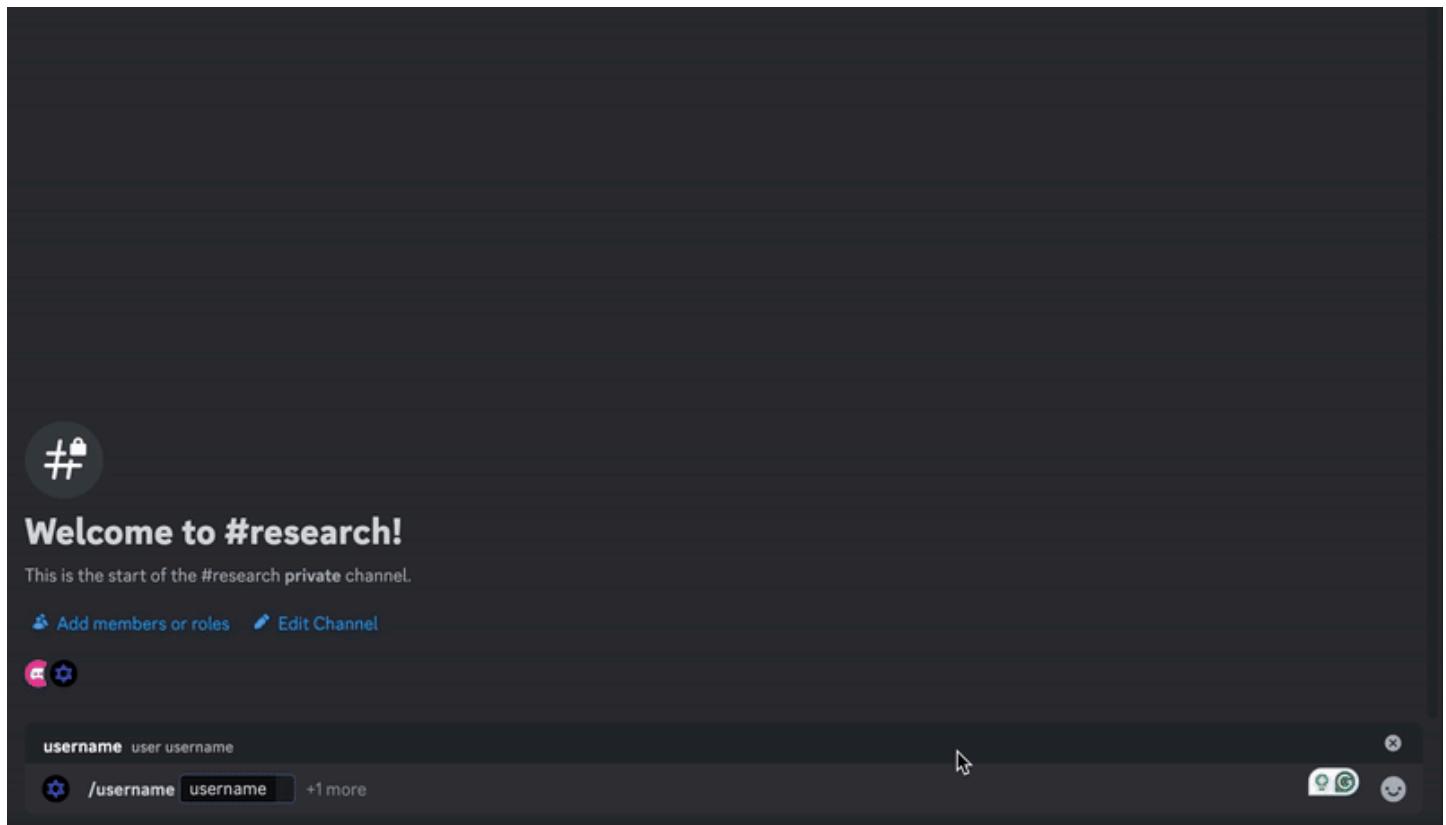
When the command is called, it uses my account credentials to access the Discord User API, changes the user avatar to a randomly generated image (to randomize the hash) and sends a friend request to a username specified. Finally, it uses a private API based on the Cloudflare Teleport CLI to run the same cache enumeration attack and displays the results directly on Discord, all in less than 30 seconds.

To show the extent of this attack, one of the first users I tried this attack on was Stanislav



Vishnevskiy, Discord's CTO.

Here's the bot in action:



The bot sends a friend request to Stan, then waits a couple of seconds to ensure he receives the push notification.

It finds 2 local Cloudflare datacenters that have cached the avatar. This could mean he has multiple devices hooked up with his Discord account that received the push notification, or his device loaded the avatar twice and the requests were load balanced within different datacenters.

GeoGuesser, powered by the Google Maps API, generates a likely location of the user. It finds the midpoint between the 2 datacenters and draws 2 circles that signify his radius.

Discord's HQ is located in San Francisco, CA (which is in the outer circle) so this map is accurate. Stan is most likely located somewhere near the edge of the inner circle which is about ~300 miles.

This entire process took less than a minute to run. I'm sure Stan saw the notification on his phone, didn't think twice and simply dismissed it. This was just a simple attack but this attack, if calibrated, can be used to track and monitor Stan's location.

An attacker like this can be launched on any Discord user and it's almost undetectable.

## Bug Bounty Reports

I responsibly disclosed to the affected parties my research, hoping something would be done to protect or warn users against this type of deanonymization attack but I was mostly disappointed.

**Signal**

Signal instantly dismissed my report, saying it wasn't their responsibility and it was up to users to hide their identity: "Signal has never attempted to fully replicate the set of network-layer anonymity features that projects like Wireguard, Tor, and other open-source VPN software can provide".

I disagree with this. Signal markets itself as a privacy-first communication platform. While it does not claim to provide network-layer anonymity like Tor, users trust Signal to minimize privacy risks.

The vulnerability demonstrates that the platform unintentionally leaks information that could narrow down a user's location within a few hundred miles. This leakage conflicts with the expectations of many privacy-conscious users who rely on Signal for more than just end-to-end encryption.

Telegram, another privacy-focused application, is completely invulnerable to this attack as (1) they use a custom in-house built protocol that's not reliant on HTTP and (2) don't rely on cloud providers like Cloudflare for caching.

## Discord

Initially, Discord's Security Team promised to look into this and make changes to protect their users against this type of attack but eventually they also changed their position on this, citing this as a Cloudflare issue other consumers are also vulnerable to.

## Cloudflare

Cloudflare ended up completing patching the bug used by Cloudflare Teleport to traverse datacenters. The bug had been reported to their HackerOne program a year ago by another reporter, but they hadn't done anything about it back then since they didn't see any impact of traversing datacenters until I shared my research.

Cloudflare reopened the old report, resolved it and awarded a \$200 bounty to my report and the original.

Although this is a step in the right direction, this doesn't actually fix the core issue here. Every attack shown in this write up has been done in the last 24 hours even though Cloudflare patched this bug weeks ago. Cloudflare patched the bug inside their network that facilitated datacenter traversal, but that's not the only way to easily traverse datacenters all over the world.

24 hours after their patch, I reprogrammed Cloudflare Teleport to use a VPN instead. Numerous VPNs provide multiple locations that users can connect to which sends their traffic through servers in different parts of the world and these servers map to different Cloudflare datacenters all over the world.

I chose a VPN provider with over 3,000 servers located in various locations across 31 different countries worldwide. Using this new method, I'm able to reach about 54% of all Cloudflare datacenters again. While this doesn't sound like a lot, this covers most places in the world with significant population.

Cloudflare's final statement about this says they do not consider the deanonymization attack to be a vulnerability in their own systems and it is up to their consumers to disable caching for resources they wish to protect.

There's clearly a problem here as Cloudflare says consumers are responsible for protecting themselves against these types of attacks, while consumers (ex. Discord) are putting the blame on Cloudflare.

## How to Protect Yourself

The potential for exploitation using this deanonymization attack is significant, especially for users in sensitive positions like journalists, activists, and privacy-conscious individuals. The attack leverages fundamental design decisions in caching and push notification systems, demonstrating how infrastructure meant to enhance performance can be misused for invasive tracking.

Although Cloudflare has patched the Teleport bug, and some applications like Discord and Signal may have implemented mitigation measures following my disclosure, the underlying risks remain. Any app using a CDN for content delivery and caching can still be vulnerable if the proper precautions aren't taken.

## Final Thought

This attack highlights how complex and interconnected the digital ecosystem has become. While CDNs improve performance and scalability, they also inadvertently introduce risks that can be exploited in novel ways. By raising awareness and promoting best practices, we can work together to minimize the potential for abuse.

For users in sensitive roles or those concerned about their privacy, the key takeaway is this: stay informed and vigilant. While no system is entirely foolproof, taking steps to limit your exposure can make a significant difference.

[Load earlier comments...](#)

 **toughyear** commented 2 days ago

great job.

 **fadhil-riyanto** commented 2 days ago

cool!



**hanliuxin5** commented 2 days ago

◻◻◻◻◻◻◻◻◻◻



**dkwo** commented 2 days ago

For Signal, is it not enough to disable Media auto-download in Data and storage (at least on Android they call it like that)?



**sinaler** commented 2 days ago • edited ▾

amazing work!



**selimozcann** commented 2 days ago • edited ▾

Amazing job that good luck for your security career



**der-ali** commented 2 days ago

Impressive work! Was fun and informative to read.



**stevefan1999-personal** commented yesterday

But a very simple way to defeat it is to just use a global VPN



**ternera** commented yesterday

cool stuff!



**hklcf** commented yesterday

Cloudflare Teleport still work?



**kyanha** commented yesterday

For Signal, is it not enough to disable Media auto-download in Data and storage (at least on Android they call it like that)?

I think mitigating the 0-click variant also requires Notifications | Messages | Show to be set to "Name only" or "No name or message".



**codl** commented yesterday

this is in no way "incredibly precise"



**gragorther** commented yesterday

Great writeup, was interesting to read. The payout is disappointing tho



**GorlikItsMe** commented yesterday

wow, thanks for sharing



**Sun-Wukong** commented yesterday

The payout is annoyingly small, but my goodness did you do your thing! A job well done on your part, much appreciated for sharing!



**dsernst** commented yesterday

There's actually a really good pattern for how to do this safely: don't do the resource loading and parsing locally for messaging apps, which is what Google does for GMail resources or for PDFs on Drive. Instead parse the content in a server-side sandbox without any kind of geolocation affinity with the client, send the client the results of this sandboxed parse.

This is essential to mitigating parser exposure in 0-click scenarios. If you are running a network like Discord or Signal, you need to act like a reverse CDN, resolving resources in your network so that your clients get required isolation.

This would break end-to-end encryption, which Gmail doesn't offer



**message** commented yesterday



 **SunsetMkt** commented yesterday

Impressive.

 **pdelteil** commented 20 hours ago

Nice bug, Impressive you have been getting bugs such as [CVE-2016-7882](#) since you were five years old. <https://hackerone.com/daniel?type=user>

Care to elaborate Daniel?

There's a bug in HackerOne that shows reports of other users incorrectly on someone else's profile. The report sent to Adobe was created by the user `daniel-hamid`, while the OP's handle is `daniel`.

Interesting! It's a f\*\*\* up bug!

 **S4GU4R0** commented 20 hours ago

It's kind of ridiculous how Cloudflare concluded in the end. Definitely should have paid you more. Thanks for sharing this and taking the time to do/write it up.

 **yyassif** commented 19 hours ago

Good job man

 **aspensmonster** commented 19 hours ago

Signal instantly dismissed my report, saying it wasn't their responsibility and it was up to users to hide their identity: "Signal has never attempted to fully replicate the set of network-layer anonymity features that projects like Wireguard, Tor, and other open-source VPN software can provide".

Not only have they never attempted it, they also seem to have no interest in doing so in the future either. I offered to start work on integrating tor routing into the app for text and data flows. Never heard anything back:

<https://community.signalusers.org/t/use-an-anonymizing-overlay-network/62670/1>

 **awwright** commented 18 hours ago

You're 15? Dude impressive work, not just the breadth of knowledge but very savvy in writing it up.

I disagree with this. Signal markets itself as a privacy-first communication platform. While it does not claim to provide network-layer anonymity like Tor, users trust Signal to minimize privacy risks.

I concur with your disagreement. The application developers are fully aware that their application is making external requests to third parties, and that this is going to leak data in a way that the users of their app are not aware. The push notification should include all of the necessary data to render the notification, without needing to make external requests.

Cloudflare's final statement about this says they do not consider the deanonymization attack to be a vulnerability in their own systems and it is up to their consumers to disable caching for resources they wish to protect.

There's clearly a problem here as Cloudflare says consumers are responsible for protecting themselves against these types of attacks, while consumers (ex. Discord) are putting the blame on Cloudflare.

Cloudflare is the correct party here. HTTP doesn't guarantee privacy, it offers many features among those privacy and caching; which are to some extent mutually exclusive. It's your responsibility as an application developer to use the features of HTTP that align with your interests. Not all CloudFlare customers need this privacy, many desire the performance.

Unfortunately there's few good solutions for this. This is essentially a traffic analysis problem; and in traffic analysis, the only foolproof workaround is to fully utilize all communication channels at all times at the maximum rate you will ever need to use.

I think there is a very important note:

The basic solution to this problem is to be considerate of side effects caused by third party actors, and be wary of all exchanges with three or more actors. In general, a third party shouldn't be able to cause me to make a network request (like requesting a favicon) that might have a side effect (like filling a cache). If necessary, that information should be attached to the push notification itself, which removes one of the three parties (the HTTP cache), meaning there's no more side effects to observe as an attacker. However requests caused just by me e.g. browsing the Web, or downloading my users list, are fine, since these are two-party exchanges and state changes to caches caused can't be correlated to me.

 **JeffreyShran** commented 13 hours ago

Good work @hackermondev!



**nQue** commented 12 hours ago

You're 15 years old...? Both the exploit and the write-up is very impressive for a 15-year old. The bug bounty was clearly way too small in this case, there should be at least one more zero at the end, though it is quite tricky for me to figure out which service provider is actually at fault here. Since there's various ways to get this information despite what they do, it's almost as if it's none of their faults, and this exploit is just due to the very nature of caching. At least there's some line of logic leading in that direction. There's lines of logic leading into all sorts of directions here. I'm doubtful of all the people saying it's clear who's at fault. I think it's very tricky! Good job anyway, kid!



**matreurai** commented 12 hours ago

Impressive mate, keep up the good work! Cheers! 🔥



**nsvk13** commented 10 hours ago

very good work! @hackermondev!



**lucasgelfond** commented 6 hours ago

Incredible work and fascinating writeup, thanks @hackermondev !



**yv-was-taken** commented 4 hours ago

banger



**xuid0** commented 1 hour ago

well done, you are a bright mind for 15 years. Don't let any payment make you think otherwise they should have paid more and given proper credit.