Signed URLs and signed cookies overview

Cloud CDN signed URLs and signed cookies let you serve responses from Google Cloud's globally distributed caches, even when you need requests to be authorized.

Cloud CDN signed URLs and signed cookies achieve similar goals: they both control access to your cached content. If you want to serve content from Google Cloud's globally distributed caches, and you're deciding between signed URLs or signed cookies, consider the following use case comparison.

Signed URLs

A signed URL is a URL that provides limited permission and time to make a request.

Use cases

In some scenarios, you might not want to require your users to have a Google Account to access Cloud CDN content, but you still want to control access by using your application-specific logic.

The typical way to address this use case is to provide a signed URL to a user, which gives the user read access to that resource for a limited time. You specify an expiration time when you create the signed URL. Anyone who knows the URL can access the resource until the expiration time for the URL is reached or the key used to sign the URL is rotated.

Use signed URLs in the following cases:

- You need to restrict access to individual files, such as an installation download.
- Your users are using client applications that don't support cookies.

How signed URLs work

Signed URLs give a client temporary access to a private resource without requiring additional authorization. To achieve this, selected elements of a request are hashed and cryptographically signed by using a strongly random key that you generate.

When a request uses the signed URL that you provided, the request is considered authorized to receive the requested content. When Cloud CDN receives a request with a bad signature for an enabled service, the request is rejected and never goes to your backend for handling.

Generally, a signed URL can be used by anyone who has it. However, a signed URL is usually only intended to be used by the client to which the URL was given. To mitigate the risk of the URL being used by a different client, signed URLs expire at a time chosen by you. To minimize the risk of a signed URL being shared, set it to expire as soon as possible.

How URLs are signed

Before you can sign URLs, you create one or more cryptographic keys on a backend service, backend bucket, or both. You then sign and cryptographically hash a URL by using the `gcloud` command-line tool or your own code.

Handling of signed URLs

When signed URL handling is enabled on a backend, Cloud CDN gives special handling to requests with signed URLs. Specifically, requests with a `Signature` query parameter are considered signed. When such a request is received, Cloud CDN verifies the following:

1. The HTTP method is `GET` or `HEAD`.
2. The `Expires` parameter is set to a future time.
3. The request's signature matches the signature computed by using the named key.

If any of these checks fails, a `403 Forbidden` response is served. Otherwise, the request is either proxied to the backend or served from the cache. All valid signed requests for a particular base url (the part before the `Expires` parameter) share the same cache entry. Responses to signed and unsigned requests do not share cache entries. Responses are cached and served until the expiration time that you set.

Content that requires signed requests is often marked as uncacheable by using the `Cache-Control` header. To make such objects compatible with Cloud CDN without requiring backend changes, Cloud CDN overrides the `Cache-Control` header when responding to requests that have valid signed URLs. Cloud CDN treats the content as cacheable, and uses the `max-age` parameter set in your Cloud CDN configuration. The response served still has the `Cache-Control` headers that the backend generated.

The URL returned from the `gcloud` command-line tool or produced by your custom code can be distributed according to your needs. We recommend signing only HTTPS URLs because HTTPS provides a secure transport that prevents the signature component of the signed URL from being intercepted. Similarly, you should distribute the signed URLs over secure transport protocols such as TLS/HTTPS.

Signed cookies

A signed cookie is a cookie that provides limited permission and time to make requests for a set of files.

Use cases

Use signed cookies in the following cases:

- You need to provide access to multiple restricted files.
- You want to avoid changing your current URLs.
- You want to avoid updating URLs each time you refresh authorization to access content.

Streaming media using HLS and DASH

If you serve video and audio content by using the HTTP Live Streaming (HLS) or Dynamic Adaptive Streaming over HTTP (DASH) protocols, you typically generate a manifest that contains a list of URLs to video and audio segments. You might have multiple instances of each segment to provide different encodings (codec, bitrate, resolution) to a client.

Although you can use Cloud CDN's signed URLs to sign and authorize access to each of these URLs, dynamically generating all possible combinations on a per-user basis is burdensome and increases origin load and application complexity.

Signed cookies are designed to address this concern. You can provide the user with a signed cookie that authorizes them to access any content that matches a policy (URL prefix and expiry date) without having to individually generate or sign your media manifests. You can refresh user access periodically through the JavaScript fetch() API on page navigation or other background mechanisms in native applications. The ability to refresh user access also lets you use short expiry times, making it harder for users to share protected content.

You can issue these cookies to users with multiple browser clients and other HTTP-speaking clients, such as Google's ExoPlayer and iOS' AVPlayer.

Binary downloads (gaming)

Similar to media streaming, if you provide game client downloads, you might divide large multi-gigabyte patches or game data into smaller chunks to support finer-grained caching, invalidation, and concurrency.

These chunks are typically listed in a manifest. Signed cookies let you authorize access to those downloads to authenticated users only without requiring modifications to the manifest, and (as with signed URLs) without foregoing the benefits of Cloud CDN caching.

How signed cookies work

Configuring and issuing signed cookies requires three steps:

- Creating a signing key for the given backend service.
- Creating a cookie value with the allowed URL prefix, expiry, key name, and cryptographic signature.
- Issuing the cookie in your application code.

Cloud CDN validates these signed cookies when they are included with requests.

You can prevent users from circumventing your signed cookie controls when using a Cloud Storage bucket. To do so, constrain access to the underlying bucket by removing the `allUsers` role and granting the Cloud CDN service account read access to the bucket.

Similarly, your virtual machine (VM) instances should validate the signatures on every signed request that they serve.

Caveats and limitations

- You are solely responsible for any consent and privacy compliance needed for your signed cookies. Signed cookies are issued and managed by you, not Google.

- If you use both signed URLs and signed cookies to control access to the same files, and a viewer uses a signed URL to request a file, Cloud CDN determines whether to return the file to the viewer based only on the signed URL. Cloud CDN only considers signed cookies if the URL is not signed.
- If you have configured your service for signed requests and your URL includes `Signature` as a query parameter, Cloud CDN attempts to interpret your URL as a signed URL. If Cloud CDN attempts to treat your URL as a signed URL when you didn't intend it, your URL likely isn't a valid signed URL, so Cloud CDN rejects it.
- Browsers and other clients typically enforce limits on cookie size (4 KB per cookie) and a total count of 50 per domain, as per RFC 6265 (https://tools.ietf.org/html/rfc6265). Consider the total cookie payload sent from their domain.
- Cloud CDN limits and restrictions apply, including a maximum of three signed request keys per backend.
- Signed requests are not charged differently from existing Cloud CDN requests. However, failed (rejected) requests, such as those with expired or otherwise invalid signatures, still incur cache lookup charges (/cdn/pricing#overview).

What's next

- To scope user access to specific URLs, see Using signed URLs (/cdn/docs/using-signed-urls).
- To scope user access to a specific URL prefix, see Using signed cookies (/cdn/docs/using-signed-cookies).

Last updated 2020-07-15 UTC.