Proxy Auth changes to address Google phishing site flagging

Author: Eugene

Background

Proxy Auth secures the Proxy against being used as an open to malicious attackers. It does so by blocking traffic until the user establishes a proxy auth session. This session can be resolved by a cookie, or in some cases by the User Session query parameter **C1f3C\_ccpContextId2\_** injected during JS rewriting.

However, in some contexts, some requests don’t support cookies and don’t have the **C1f3C\_ccpContextId2\_** query parameter (not originated from JS-rewritten code).

In such cases, the proxy performs the Uncredentialed Request Protocol to determine whether the client is capable of handling cookies. If the client proves that it is capable of handling cookies, it is put through the rest of the protocol. Otherwise, the endpoint is deemed to be an Uncredentialed Context endpoint, cached in the Uncredentialed Endpoint (UE) Cache, and allowed to go through to the cloud, under the assumption that this endpoint must be for requests made in uncredentialed context (e.g. an XHR or Fetch request with *credentials: false*). The next time an cookieless request comes to the same URL (not including query parameters), it is also treated as uncredentialed if it is present in the UE Cache, to improve performance of handling multiple uncredentialed requests to the same endpoint. If at any point a request with a cookie comes to the same endpoint, we remove the endpoint from the UE Cache, assuming that this endpoint may be common for credentialed and uncredentialed requests, and thus cannot be optimized.

# Problem

Google crawls our domains using a [Netcraft](https://www.netcraft.com/) crawler. Apparently, this crawler doesn’t support cookies, so every endpoint ends up being treated as an uncredentialed request. So we allow proxying the front page of Google.com:

we fetched <https://accounts-google-com-support.ciphercloud.io/ServiceLogin> and were ultimately redirected to <https://accounts-google-com-support.ciphercloud.io/signin/v2/identifier?flowName=GlifWebSignIn&flowEntry=ServiceLogin> where we we were served a google login page proxied by ciphercloud.

We suggest adding simple text such as: "CipherCloud is proxying this page from login.google.com."

That would explain the situation to users, and is in line with our public policy language related to third-party services. (<https://developers.google.com/search/docs/advanced/security/social-engineering#what-is-social-engineering> )

There’s also a possibility of temporary UE Cache poisoning by such uncredentialed requests that may impact future requests to the same endpoint that are missing cookies. A request that contains cookies will fix the issue.

# Proxy Auth protocol changes

We discussed several changs we can make to the Proxy Protocol around Uncredentialed request processing. We want to prevent user agents from being able to proxy through Proxy Auth with uncredentialed requests until we’ve seen them authenticate at least once (proving that they are capable of authentication), unless they’re explicitly whitelisted in the User Agent Whitelist.

1. We will use (IP + User Agent) to add some verification that the user agent has been previously authenticated into the Proxy. Otherwise, we don’t allow uncredentialed requests.
2. We can also consider including IP and/or User Agent information in the UE cache in addition to the endpoint. However, it seems less important if we implement (1), as sessionless User Agents will be blocked, and cookie-based User Agents are expected to access the endpoints consistently.
3. For logging purposes, we will maintain counts of unique Proxy Auth users coming from the same IP address
4. We will check feasibility if we can completely remove Uncredentialed request processing flow and rely on User-Agents to always provide uncredentialed requests with the **C1f3C\_ccpContextId2\_** query parameter, otherwise require whitelisting.

# Proxy Banner

Per Google recommendation, we will add a banner for the new users **only in the Reverse Proxy mode**. The banner should go to every (top Window) page served by the proxy until the user dismisses it (similar to Accept Cookies… banners). When the user dismisses the banner, we will set a cookie in the RP domain to remember this, and not show it again after that.

We should add the banner using JS injected as part of JS rewriting logic. Apps that don’t evaluate JS on the page would not be affected.