Web Cache Poisoning Vulnerability Analysis - Figma

Background:

Web cache poisoning is an attack where a malicious request is stored in the cache, and unsuspecting users are served the malicious content from the cache instead of the legitimate one. This can lead to malicious payloads being delivered to multiple users without their knowledge, exploiting vulnerabilities in how requests and responses are handled.

Vulnerability Details:

The analysis of Figma's HTTP response headers reveals several cookies and cache-related directives that may contribute to potential web cache poisoning attacks. Below is a snippet of a typical HTTP response from Figma:

HTTP Response Example:

HTTP/1.1 200 OK
Date: Sat, 19 Oct 2024 23:13:38 GMT
Content-Type: text/html; charset=utf-8
Connection: close
x-sorting-hat-podid: 370
x-sorting-hat-shopid: 57683640503
x-storefront-renderer-rendered: 1
set-cookie: keep_alive=507cfa2a-7a6c-48b2-b6f5-2a0e0893708d; path=/; expires=Sat, 19 Oct 2024 23:43:38 GMT; HttpOnly; SameSite=Lax
set-cookie: secure_customer_sig=; path=/; expires=Sun, 19 Oct 2025 23:13:38 GMT; secure; HttpOnly; SameSite=Lax
set-cookie: localization=IN; path=/; expires=Sun, 19 Oct 2025 23:13:38 GMT; SameSite=Lax
set-cookie: cart_currency=INR; path=/; expires=Sat, 02 Nov 2024 23:13:38 GMT; SameSite=Lax
set-cookie: _cmp_a=%7B%22purposes%22%3A%7B%22a%22%3Atrue%2C%22p%22%3Atrue%2C%22m%22%3Atrue%2C%22t%22%3Atrue%7D%2C%22display_banner%22%3Afalse%2C%22sale_of_data_region%22%3Afalse%7D; domain=figma.com; path=/; expires=Sun, 20 Oct 2024 23:13:38 GMT; SameSite=Lax
set-cookie: _shopify_y=bbb3dbf4-f4ce-4b44-8743-50d35816648a; Expires=Sun, 19-Oct-25 23:13:38 GMT; Domain=figma.com; Path=/; SameSite=Lax
set-cookie: _shopify_s=505c2ea3-fc42-4c20-9f61-7b1269c7d38c; Expires=Sat, 19-Oct-24 23:43:38 GMT; Domain=figma.com; Path=/; SameSite=Lax
set-cookie: receive-cookie-deprecation=1; Secure; HttpOnly; SameSite=None; Path=/; Partitioned;

x-shopify-nginx-no-cookies: 0
link: <https://cdn.shopify.com>; rel="preconnect", <https://cdn.shopify.com>; rel="preconnect"; crossorigin
etag: "cacheable:9174211e072c4b60bcfea658038d510a"
x-cache: miss
x-frame-options: DENY
content-security-policy: block-all-mixed-content; frame-ancestors 'none'; upgrade-insecure-requests;
access-control-allow-origin: *
strict-transport-security: max-age=7889238
x-shopid: 57683640503
x-shardid: 370
vary: Accept
content-language: en-IN
powered-by: Shopify
server-timing: processing;dur=171;desc="gc:47", db;dur=14, db_async;dur=6.264, parse;dur=3, render;dur=55, asn;desc="24186", edge;desc="HYD", country;desc="IN", theme;desc="172070076787", pageType;desc="product", servedBy;desc="grrx", requestID;desc="76b7220d-3159-4de0-9950-a8fe7e924fcb-1729379618"
x-dc: gcp-asia-southeast1,gcp-asia-southeast1,gcp-asia-southeast1
x-request-id: 76b7220d-3159-4de0-9950-a8fe7e924fcb-1729379618
Alt-Svc: h3=":443"; ma=86400
CF-Cache-Status: DYNAMIC
Server-Timing: cfRequestDuration;dur=263.999939
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Permitted-Cross-Domain-Policies: none
X-Download-Options: noopen
Server: cloudflare
CF-RAY: 8d5475b8792679ef-HYD
Content-Length: 259949

Key Security Observations:

1. x-cache: miss Header:

The presence of the x-cache: miss header indicates that this response was not served from cache but was fetched from the origin server. This suggests that the caching mechanism is active but didn't serve this particular request from cache.

Risk: If a malicious response is stored in the cache for future requests, it may serve harmful content to other users.

2. Content Security Policy (CSP):

The CSP header blocks mixed content and forbids framing (frame-ancestors 'none'), which strengthens protection against certain types of cross-site scripting (XSS) attacks.

Mitigation: This helps mitigate some risks of serving malicious content, but the focus should still be on controlling the caching behavior.

3. Vary Header:

The vary: Accept header specifies that the cached response depends on the Accept header of the request. This is a positive approach to prevent caching of responses that should not be universally applied.

Risk: Misconfigurations in this header could allow poisoning of cache if different types of requests result in cached malicious responses.

4. Cookies:

Multiple cookies (secure_customer_sig, keep_alive, etc.) are set with attributes like HttpOnly and SameSite=Lax, providing some degree of security against cross-site scripting (XSS) and cross-site request forgery (CSRF).

Risk: If cookies are not appropriately handled in cache mechanisms, attackers could manipulate cookie values to persist malicious sessions in the cache.

5. Etag Header:

The etag header used for caching can be leveraged in cache poisoning if the response from the server and cached versions are not synchronized.

Risk: Attackers could manipulate Etags to inject malicious responses into the cache.

Recommendations:

1. Strengthen Cache Control Mechanisms:

Implement stricter cache-control headers to prevent caching sensitive or user-specific content. Headers like Cache-Control: no-store should be considered for dynamic content or user-specific data.

2. Enhance Vary Header Configuration:

Ensure that the Vary header is correctly configured to handle different types of user requests and responses, reducing the risk of cache poisoning due to incorrect caching.

3. Regular Audits:

Continuously audit and monitor caching behaviors across different endpoints, particularly where user-generated content or dynamic responses are involved.

4. Mitigate Risks with Web Application Firewalls (WAFs):

Consider deploying a WAF that can monitor and filter potentially malicious requests, blocking attempts to poison the cache.
