Remediation Playbook

Generated 23rd Nov 2026 at 10:09 (UTC)

Remediation Playbook: Cacheable HTTPS Response — ian org child asset-62

Vulnerability: Cacheable HTTPS response — Cacheable HTTPS Response on `altoromutual.com`. HTTPS responses containing sensitive content are not instructed to prevent local browser caching, meaning any user on a shared or compromised device could retrieve sensitive application data from the browser cache without re-authenticating. The remediation deadline is **2018-02-28** (date opened 2017-08-02 plus 180 days, default window for Severity 1 / Minimal risk; no `sla_violation_date` is present in your data). **This vulnerability is significantly overdue — contain and remediate immediately.**

> ⚠️ No CVEs are associated with this finding, `on_cisa_list` is false, and the vulnerability does not permit remote code execution. The Compromise Assessment phase is therefore omitted, and phases are numbered 1–4 accordingly.

---

1. Validate & Scope — by 2026-11-25 🔍

1. **Confirm the finding is still present.** Send a request to `altoromutual.com` over HTTPS and inspect the response headers for `Cache-Control` and `Pragma` directives. The absence of `no-store` / `no-cache` confirms the finding.

   ```
   curl -sI https://altoromutual.com/ | grep -i -E "cache-control|pragma|expires"
   ```

   A vulnerable response will return no `Cache-Control: no-store` or will show permissive values such as `public`, `max-age`, or no header at all.

2. **Enumerate affected paths.** Identify all endpoints that return sensitive content (authenticated pages, account data, form responses, API responses). Repeat the header check across a representative sample:

   ```
   for PATH in / /login /account /profile /api/; do
     echo "--- $PATH ---"
     curl -sI "https://altoromutual.com${PATH}" | grep -i -E "cache-control|pragma|expires"
   done
   ```

3. **Record exposure.**
   - **Exposure:** Internet-facing; asset is externally reachable via HTTPS (port 443 implied by the HTTPS context; no additional ports are recorded in your data).
   - **Layer:** Application (Layer 7).
   - **Protocol:** HTTPS.
   - **Host:** `altoromutual.com` (ian org child asset-62).

4. **Identify the likely owner.** The asset is recorded as a child asset within the ian org. The web application and server configuration owner should be identified from the internal CMDB or asset register. **Assumption:** ownership has not been explicitly recorded in your data; confirm the responsible application/infrastructure team before proceeding.
5. **Identify dependent systems.** Determine whether a reverse proxy, CDN, or load balancer sits in front of `altoromutual.com` — caching headers may need to be set at that layer as well as at the origin server. **Assumption:** no CDN or intermediary caching layer has been confirmed from your data; verify this during scoping.

---

2. Contain — by 2026-11-26 🛡️

1. **Apply a WAF virtual patch.** Because this vulnerability is web-addressable, the **WAF Virtual Patch** insight within the Edgescan portal can generate a virtual-patch rule that injects the required `Cache-Control` and `Pragma` response headers at the WAF layer for `altoromutual.com` before the permanent server-side fix is in place. Enable this rule as an interim measure.
2. **Inject cache-control headers at the web server or reverse proxy level (interim).** If a WAF is not immediately available, add the headers at the nearest network-edge layer. For an Nginx reverse proxy:

```
# Interim: add to the server {} block for altoromutual.com
add_header Cache-Control "no-store" always;
add_header Pragma "no-cache" always;
```

For Apache (interim `.htaccess` or `VirtualHost`):

```
<IfModule mod_headers.c>
    Header always set Cache-Control "no-store"
    Header always set Pragma "no-cache"
</IfModule>
```

Reload the service after any interim change:

```
# Nginx
nginx -t && systemctl reload nginx

# Apache
apachectl configtest && systemctl reload apache2
```

3. **Notify stakeholders.** Inform the application owner, security team, and change management of the interim controls and the overdue status of this finding, referencing the NIST SP 800-61r3 communication guidance for open vulnerabilities.
4. **Restrict access to cached content on shared endpoints (compensating control).** Where shared or kiosk devices are known to access `altoromutual.com`, enforce a policy (via endpoint management tooling) to disable persistent browser caching or apply private/incognito mode requirements until the permanent fix is deployed.

---

3. Remediate — by 2026-11-30 (overdue; begin immediately) 🔧

> **Deadline source:** Default window of 180 days from `date_opened` (2017-08-02), applicable to Severity 1 / Minimal risk. No `sla_violation_date` is present. The calculated deadline (2018-02-28) has long passed; remediation is critically overdue.

1. **Take a configuration backup / rollback point** before making any changes.

```
# Example for Nginx
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak-$(date +%Y%m%d%H%M%S)
cp -r /etc/nginx/sites-available/ /etc/nginx/sites-available.bak-$(date
+%Y%m%d%H%M%S)/

# Example for Apache
cp /etc/apache2/apache2.conf /etc/apache2/apache2.conf.bak-$(date
+%Y%m%d%H%M%S)
cp -r /etc/apache2/sites-available/ /etc/apache2/sites-
available.bak-$(date +%Y%m%d%H%M%S)/
```

2. **Apply the permanent fix at the web server.** Set `Cache-Control: no-store` and `Pragma: no-cache` on all responses that carry sensitive content. The directives below cover all responses site-wide; scope them to sensitive paths if a blanket policy is unsuitable.

Nginx (permanent, `server {}` block):

```
server {
    listen 443 ssl;
    server_name altoromutual.com;

    # Permanent cache-prevention headers for all responses
    add_header Cache-Control "no-store" always;
    add_header Pragma "no-cache" always;

    # ... remainder of server configuration ...
}
```

Apache (permanent, `VirtualHost` or `.htaccess`):

```
<IfModule mod_headers.c>
    Header always set Cache-Control "no-store"
    Header always set Pragma "no-cache"
</IfModule>
```

IIS (via `web.config`):

```
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="Cache-Control" value="no-store" />
        <add name="Pragma" value="no-cache" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

3. **Apply at the application layer for dynamically generated sensitive responses.** For server-side code that generates authenticated or sensitive output, set headers programmatically to ensure coverage regardless of server configuration:

Python (Flask example):

```
@app.after_request
def set_cache_headers(response):
    response.headers["Cache-Control"] = "no-store"
    response.headers["Pragma"] = "no-cache"
    return response
```

Java (Servlet Filter example):

```
response.setHeader("Cache-Control", "no-store");
response.setHeader("Pragma", "no-cache");
```

4. **If a CDN or reverse proxy is confirmed upstream,** apply equivalent cache-prevention rules at that layer to ensure no sensitive content is cached at the edge. The specific mechanism will depend on the CDN vendor's configuration interface. **Assumption:** confirm CDN presence during scoping (Phase 1, Step 5).
5. **Reload / restart the web server and confirm the change is live:**

```
# Nginx
nginx -t && systemctl reload nginx

# Apache
apachectl configtest && systemctl reload apache2

# IIS
iisreset /noforce
```

6. **Remove interim WAF virtual-patch rules** once the permanent server-side fix has been verified, to avoid redundant rule overhead.

---

4. Verify & Harden — by 2026-12-07 ✅

1. **Confirm headers are present on all sensitive paths.** Run the following checks against the live service after the fix is applied:

```
curl -sI https://altoromutual.com/ | grep -i -E "cache-control|pragma"
# Expected output:
# Cache-Control: no-store
# Pragma: no-cache
```

Repeat for every sensitive endpoint identified in Phase 1, Step 2.

2. **Verify no permissive intermediary overrides the headers.** If a CDN or proxy is in the path, confirm that `Cache-Control: no-store` reaches the browser client by inspecting headers from outside the internal network:

```
curl -sI --resolve altoromutual.com:443:<external-ip>
https://altoromutual.com/account \
  | grep -i -E "cache-control|pragma|x-cache"
```

> **Assumption:** `<external-ip>` should be substituted with the confirmed public IP of `altoromutual.com` from your DNS records.

3. **Check browser cache behaviour manually.** Access a sensitive page on `altoromutual.com` in a browser, then navigate to the browser's cached resources (e.g., `about:cache` in Firefox, `chrome://cache/` in Chromium) and confirm the sensitive response is not stored.
4. **Request an Edgescan retest.** Submit a retest request for vulnerability Cacheable HTTPS response on asset ian org child asset-62 (`altoromutual.com`) via the Edgescan portal to obtain a verified close of this finding.
5. **Harden the full HTTP security header posture.** Use the opportunity to audit and enforce additional security response headers across `altoromutual.com` :

```
# Recommended additional hardening headers (Nginx example)
add_header X-Content-Type-Options "nosniff" always;
add_header X-Frame-Options "DENY" always;
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains;
preload" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Permissions-Policy "geolocation=(), microphone=(), camera=()"
always;
```

6. **Integrate header validation into the CI/CD pipeline.** Add a post-deployment test that checks for `Cache-Control: no-store` on sensitive endpoints, failing the pipeline if the header is absent:

```
HEADER=$(curl -sI https://altoromutual.com/ | grep -i "cache-control")
if [[ "$HEADER" != *"no-store"* ]]; then
  echo "FAIL: Cache-Control: no-store header missing"
  exit 1
fi
echo "PASS: Cache-Control header present"
```

7. **Establish a periodic review cadence.** Per NIST SP 800-40r4, schedule quarterly checks of HTTP response headers across all externally facing assets to detect configuration drift.

References 📚

Reference	Details
NIST SP 800-40r4	*Guide to Enterprise Patch Management Planning* — patch prioritisation and remediation workflow
NIST SP 800-61r3	*Computer Security Incident Handling Guide* — stakeholder communication and response phases
CISA Vulnerability Response Playbook	Validation, containment, and remediation sequencing — https://www.cisa.gov/known-exploited-vulnerabilities-catalog
OWASP Cheat Sheet: **HTTP Security Response Headers Cheat Sheet**	Canonical guidance on `Cache-Control` , `Pragma` , and related headers — https://cheatsheetseries.owasp.org/
OWASP ASVS	**Chapter V14 — Configuration** covers server-side security header requirements

Vendor security guidance	Consult the top-level security pages for the web server in use: Nginx — https://nginx.org/en/docs/, Apache — https://httpd.apache.org/security/, IIS — https://www.iis.net/

Request a retest via the API

Once the fix is live, request an Edgescan retest of this vulnerability without logging in to the portal. Authenticate with an API token in the `X-API-TOKEN` header (creating API tokens).

```
curl -X PUT "https://preprod.edgescan.com/api/v1/vulnerabilities/retest.json" \
  -H "X-API-TOKEN: $EDGESCAN_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"vulnerability_ids": [28841]}'
```

The response lists the vulnerability under `retesting` or `queued`, or says under `refused` why it cannot be retested (with status 400 when nothing was accepted). An id the token cannot see returns 400 with `errors` instead. Add `"preview": true` to the body to check without requesting the retest.