# Sri Lanka Institute of Information Technology

## IE2062 - Web Security

## Final Assignment

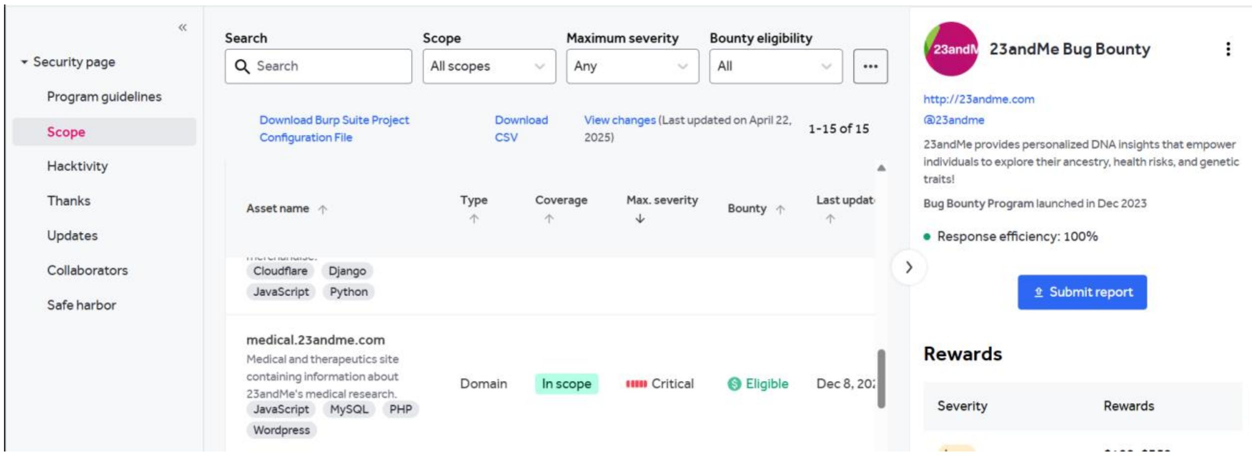
## Bug Bounty Report 03

**Name:** Peiris W.S.S.N

**Registration Number:** IT23227286

## Contents

## 1. Introduction



(Last updated on April 22, 2025) 1-15 of 15

| Asset name | Type | Coverage | Max. severity | Bounty | Last updat |
|---|---|---|---|---|---|
| Cloudflare Django JavaScript Python | | | | | |
| medical.23andme.com Medical and therapeutics site containing information about 23andMe's medical research. JavaScript MySQL PHP Wordpress | Domain | In scope | Critical | Eligible | Dec 8, 20: |

23andMe Bug Bounty

http://23andme.com
@23andme

23andMe provides personalized DNA insights that empower individuals to explore their ancestry, health risks, and genetic traits!

Bug Bounty Program launched in Dec 2023

Response efficiency: 100%

Rewards

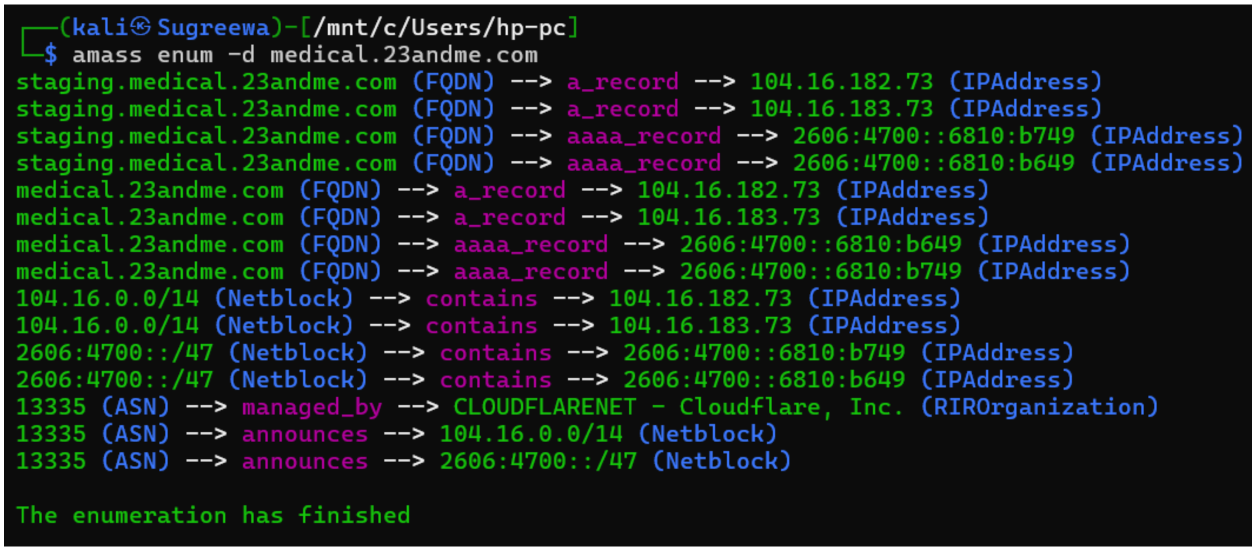
| Severity | Rewards |
|---|---|

**Website:** https://medical.23andme.com/

**Listed by:** 23andMe Bug Bounty

## 2. Reconnaissance

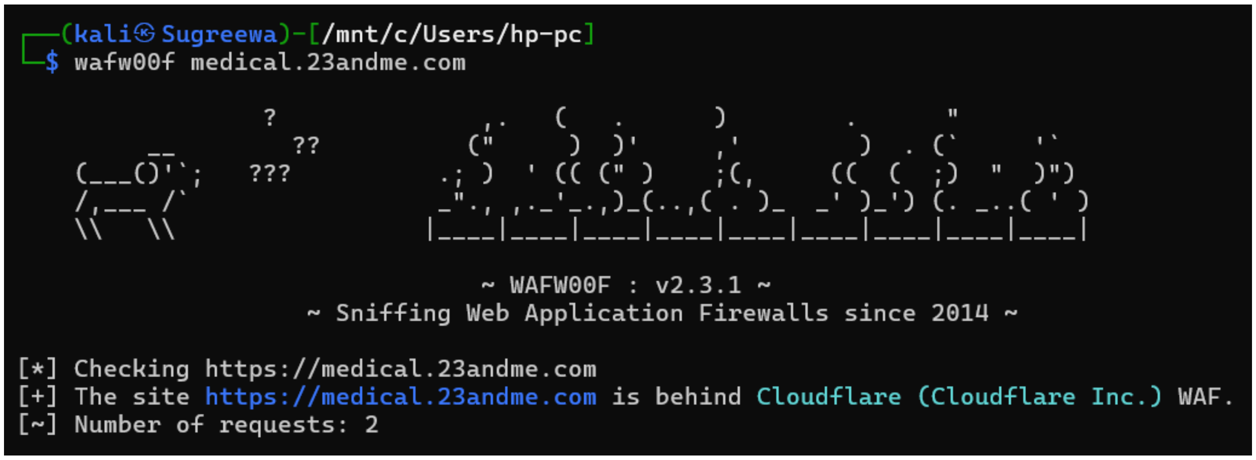
- **Subdomain enumeration using Amass**



```
┌──(kali㉿Sugreewa)-[/mnt/c/Users/hp-pc]
└─$ amass enum -d medical.23andme.com
staging.medical.23andme.com (FQDN) --> a_record --> 104.16.182.73 (IPAddress)
staging.medical.23andme.com (FQDN) --> a_record --> 104.16.183.73 (IPAddress)
staging.medical.23andme.com (FQDN) --> aaaa_record --> 2606:4700::6810:b749 (IPAddress)
staging.medical.23andme.com (FQDN) --> aaaa_record --> 2606:4700::6810:b649 (IPAddress)
medical.23andme.com (FQDN) --> a_record --> 104.16.182.73 (IPAddress)
medical.23andme.com (FQDN) --> a_record --> 104.16.183.73 (IPAddress)
medical.23andme.com (FQDN) --> aaaa_record --> 2606:4700::6810:b649 (IPAddress)
medical.23andme.com (FQDN) --> aaaa_record --> 2606:4700::6810:b749 (IPAddress)
104.16.0.0/14 (Netblock) --> contains --> 104.16.182.73 (IPAddress)
104.16.0.0/14 (Netblock) --> contains --> 104.16.183.73 (IPAddress)
2606:4700::/47 (Netblock) --> contains --> 2606:4700::6810:b749 (IPAddress)
2606:4700::/47 (Netblock) --> contains --> 2606:4700::6810:b649 (IPAddress)
13335 (ASN) --> managed_by --> CLOUDFLARENET - Cloudflare, Inc. (RIROrganization)
13335 (ASN) --> announces --> 104.16.0.0/14 (Netblock)
13335 (ASN) --> announces --> 2606:4700::/47 (Netblock)

The enumeration has finished
```

- **Firewall Detection**



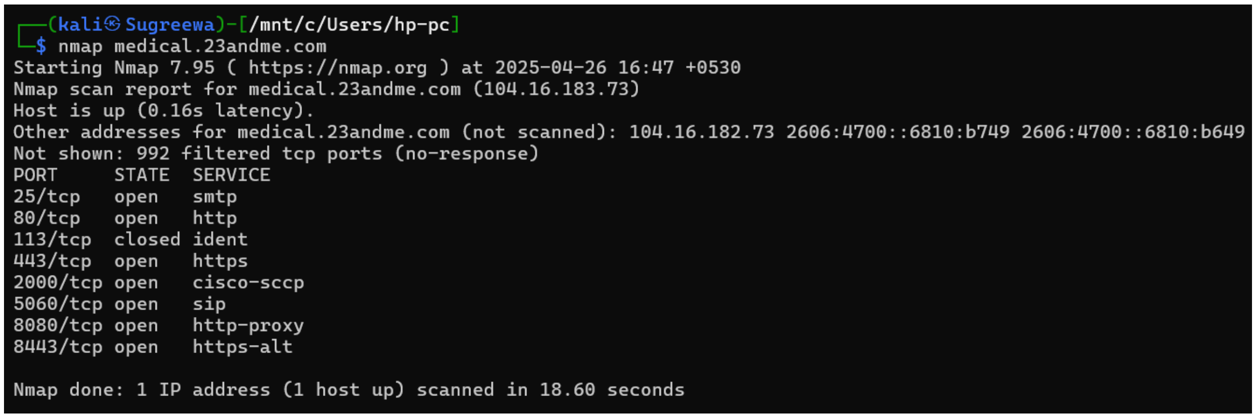
```
┌──(kali㉿Sugreewa)-[/mnt/c/Users/hp-pc]
└─$ wafw00f medical.23andme.com

                   ?              ,.   (   .      )        .      "
           __          ??          ("     )  )'     ,'        )  . (`     '`
    (___()'`;   ???          .; )  ' (( (" )    ;(,     ((  (  ;)  "  )")
    /,___ /`                 _"., ,._'_.,)_(..,( . )_  _' )_') (. _..( ' )
    \\   \\                 |____|____|____|____|____|____|____|____|____|

                            ~ WAFW00F : v2.3.1 ~
              ~ Sniffing Web Application Firewalls since 2014 ~

[*] Checking https://medical.23andme.com
[+] The site https://medical.23andme.com is behind Cloudflare (Cloudflare Inc.) WAF.
[~] Number of requests: 2
```

- **Nmap Scan**

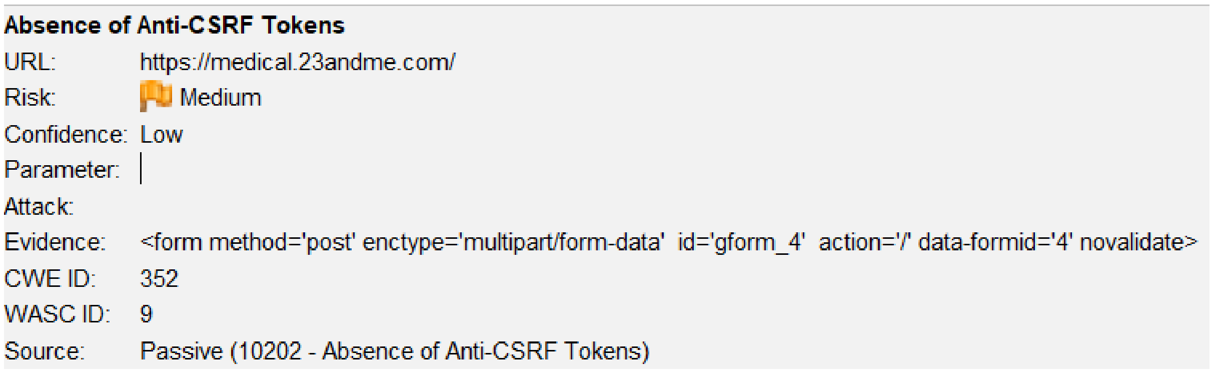


```
┌──(kali㉿Sugreewa)-[/mnt/c/Users/hp-pc]
└─$ nmap medical.23andme.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-26 16:47 +0530
Nmap scan report for medical.23andme.com (104.16.183.73)
Host is up (0.16s latency).
Other addresses for medical.23andme.com (not scanned): 104.16.182.73 2606:4700::6810:b749 2606:4700::6810:b649
Not shown: 992 filtered tcp ports (no-response)
PORT     STATE  SERVICE
25/tcp   open   smtp
80/tcp   open   http
113/tcp  closed ident
443/tcp  open   https
2000/tcp open   cisco-sccp
5060/tcp open   sip
8080/tcp open   http-proxy
8443/tcp open   https-alt

Nmap done: 1 IP address (1 host up) scanned in 18.60 seconds
```

## 3. Vulnerability

- **Absence of Anti-CSRF Tokens**



**Absence of Anti-CSRF Tokens**

| | |
|---|---|
| URL: | https://medical.23andme.com/ |
| Risk: | Medium |
| Confidence: | Low |
| Parameter: | |
| Attack: | |
| Evidence: | <form method='post' enctype='multipart/form-data' id='gform_4' action='/' data-formid='4' novalidate> |
| CWE ID: | 352 |
| WASC ID: | 9 |
| Source: | Passive (10202 - Absence of Anti-CSRF Tokens) |

## 4. Vulnerability description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

- CSRF attacks are effective in a number of situations, including:
- The victim has an active session on the target site.
- The victim is authenticated via HTTP auth on the target site.
- The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

## 5. Affected Components

- **Form ID:** gform_4
- **Form Method:** POST
- **Endpoint:** /
- **Content Type:** multipart/form-data
- **Missing Security Elements:** No known anti-CSRF token detected (e.g., csrf_token, _csrf, CSRFToken, etc.)
- **Form Fields Involved:** gform_field_values, gform_submit, gform_submit_button_4, input_4_1, state_4, etc.

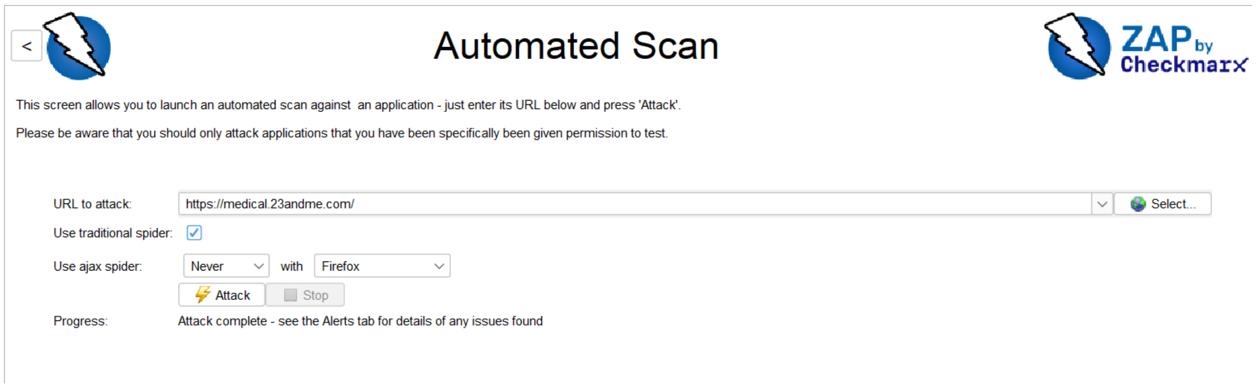
## 6. Impact Assessment

**Risk Level:** Medium

**Impacts:**

- Unauthorized transactions or actions (e.g., form submissions, state changes)
- Data leakage or manipulation
- Account hijacking (if combined with XSS)
- Potential for privilege escalation and session abuse

## 7. Steps to reproduce


Automated Scan

This screen allows you to launch an automated scan against  an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

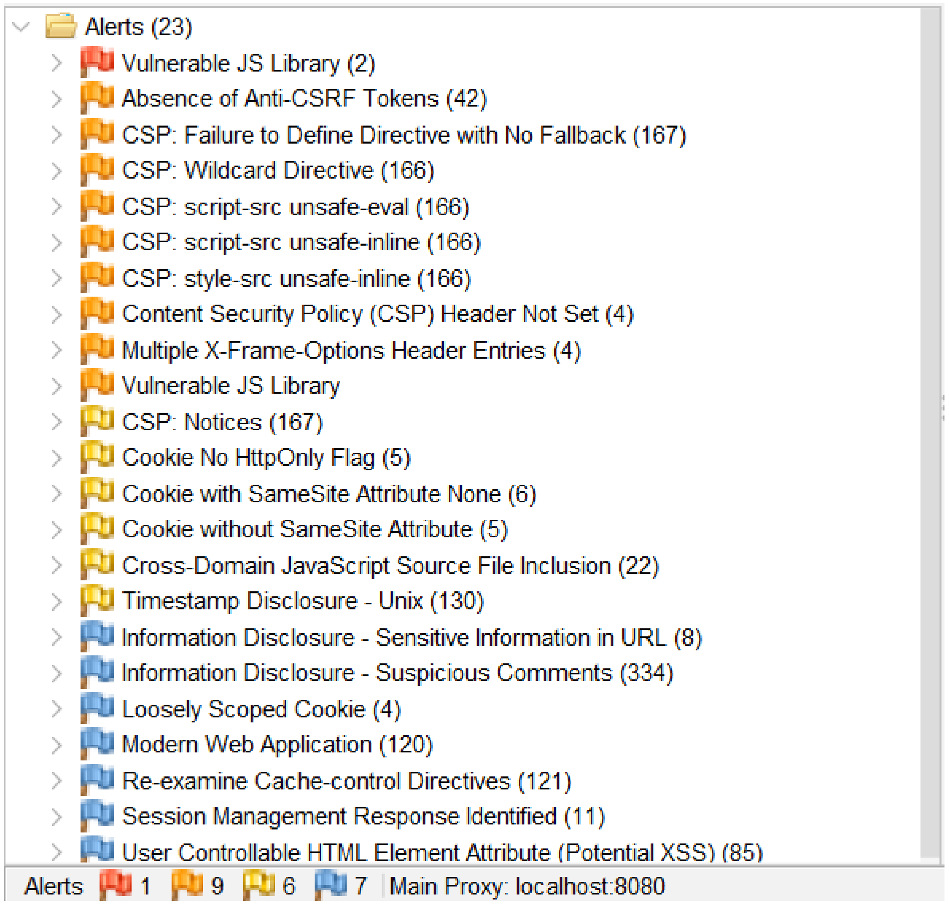
URL to attack: https://medical.23andme.com/

Use traditional spider: ✓

Use ajax spider: Never with Firefox

Progress: Attack complete - see the Alerts tab for details of any issues found

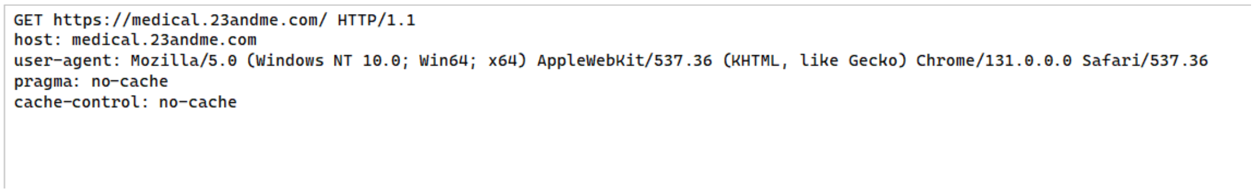
Perform a scan with Zap's automated scan and go to the alerts tab.


Alerts (23)

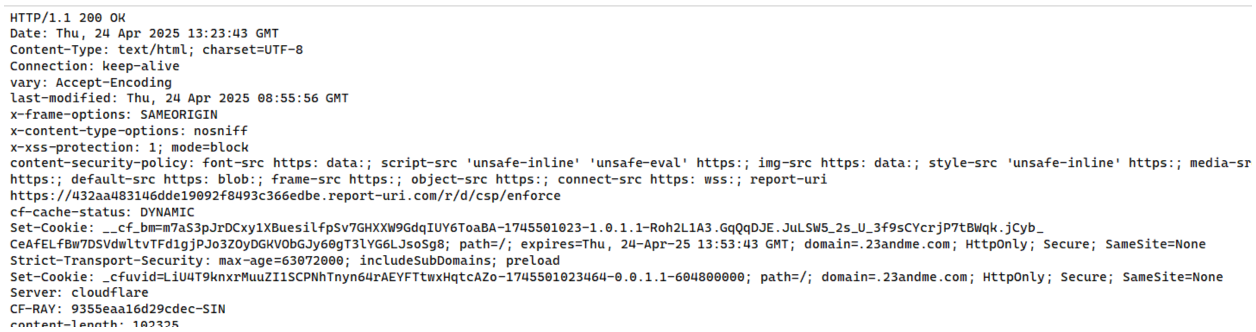
- Vulnerable JS Library (2)
- Absence of Anti-CSRF Tokens (42)
- CSP: Failure to Define Directive with No Fallback (167)
- CSP: Wildcard Directive (166)
- CSP: script-src unsafe-eval (166)
- CSP: script-src unsafe-inline (166)
- CSP: style-src unsafe-inline (166)
- Content Security Policy (CSP) Header Not Set (4)
- Multiple X-Frame-Options Header Entries (4)
- Vulnerable JS Library
- CSP: Notices (167)
- Cookie No HttpOnly Flag (5)
- Cookie with SameSite Attribute None (6)
- Cookie without SameSite Attribute (5)
- Cross-Domain JavaScript Source File Inclusion (22)
- Timestamp Disclosure - Unix (130)
- Information Disclosure - Sensitive Information in URL (8)
- Information Disclosure - Suspicious Comments (334)
- Loosely Scoped Cookie (4)
- Modern Web Application (120)
- Re-examine Cache-control Directives (121)
- Session Management Response Identified (11)
- User Controllable HTML Element Attribute (Potential XSS) (85)

Alerts 1 9 6 7 Main Proxy: localhost:8080

## 8. Proof of concept



```
GET https://medical.23andme.com/ HTTP/1.1
host: medical.23andme.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
```



```
HTTP/1.1 200 OK
Date: Thu, 24 Apr 2025 13:23:43 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
vary: Accept-Encoding
last-modified: Thu, 24 Apr 2025 08:55:56 GMT
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
content-security-policy: font-src https: data:; script-src 'unsafe-inline' 'unsafe-eval' https:; img-src https: data:; style-src 'unsafe-inline' https:; media-sr
https:; default-src https: blob:; frame-src https:; object-src https:; connect-src https: wss:; report-uri
https://432aa483146dde19092f8493c366edbe.report-uri.com/r/d/csp/enforce
cf-cache-status: DYNAMIC
Set-Cookie: __cf_bm=m7aS3pJrDCxy1XBuesilfpSv7GHXXW9GdqIUY6ToaBA-1745501023-1.0.1.1-Roh2L1A3.GqQqDJE.JuLSW5_2s_U_3f9sCYcrjP7tBWqk.jCyb_
CeAfELfBw7DSVdwltvTFd1gjPJo3ZOyDGKVObGJy60gT3lYG6LJsoSg8; path=/; expires=Thu, 24-Apr-25 13:53:43 GMT; domain=.23andme.com; HttpOnly; Secure; SameSite=None
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
Set-Cookie: _cfuvid=LiU4T9knxrMuuZI1SCPNhTnyn64rAEYFTtwxHqtcAZo-1745501023464-0.0.1.1-604800000; path=/; domain=.23andme.com; HttpOnly; Secure; SameSite=None
Server: cloudflare
CF-RAY: 9355eaa16d29cdec-SIN
content-length: 102325
```

## 9. Proposed mitigation or fix

**Phase: Architecture and Design**

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard.

**Phase: Implementation**

Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

**Phase: Architecture and Design**

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS. Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS.

Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change.

**Phase: Implementation**

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.