

# VULNERABILITY REPORT

APIsec Security Assessment

**Application:** 20663 - DST - SPIR 3359  
**Host URL:** https://blue.www.olb-qa10.dev.bmo.com  
**Scan ID:** 019b1457-62e6-7df1-9270-00cfcc563020  
**Status:** Complete  
**Generated:** December 17, 2025 at 05:24 PM

## Table of Contents

Executive Summary	2
Scan Statistics	2
Vulnerability Summary	2
Endpoints Scanned	2
Vulnerabilities	3
Informational Findings	8

# Executive Summary

This scan identified 3 vulnerabilities and 40 informational findings.



## Scan Statistics

Endpoints Scanned: 21 / 21	Total Tests: 2079	Tests Passed: 881
Tests Failed: 7	Total Findings: 43	Endpoints Affected: 20

## Vulnerability Summary

Severity	Detection ID	Test Name	Endpoint	CVSS
MEDIUM	12e23285	incremental	post /launch-openaccount/services...	5.0
MEDIUM	1d43c389	secure	get /launch-openaccount/services/...	5.0
MEDIUM	94651643	secure	get /launch-openaccount/services/...	5.0

## Endpoints Scanned

Method	Endpoint	Vulns	Info	Auth
post	/launch-openaccount/services/party-data-management/party-profile	1	6	Yes
get	/launch-openaccount/services/product-sales-agreement/everyday-banking-...	1	5	Yes
get	/launch-openaccount/services/session-dialogue/onboarding-journey/init	1	2	Yes
post	/launch-openaccount/services/party-data-management/party-profile/addres...	0	3	Yes
post	/launch-openaccount/services/processing-order/id-verification-and-credit-ris...	0	2	Yes
post	/launch-openaccount/services/processing-order/kyc-evaluation-and-credit-ri...	0	2	Yes
post	/launch-openaccount/services/product-sales-agreement/everyday-banking-...	0	2	Yes
post	/launch-openaccount/services/product-sales-agreement/everyday-banking-...	0	2	Yes
patch	/launch-openaccount/services/product-sales-agreement/everyday-banking-...	0	2	Yes
post	/launch-openaccount/services/session-dialogue/onboarding-journey/deliver...	0	2	Yes
post	/launch-openaccount/services/session-dialogue/onboarding-journey/registe...	0	2	Yes
get	/launch-openaccount/services/session-dialogue/onboarding-journey/status	0	2	Yes
post	/launch-openaccount/services/session-dialogue/prospect-product-selection/...	0	2	Yes
get	/session-dialogue/onboarding-journey/end	0	2	Yes
post	/session-dialogue/onboarding-journey/status	0	2	Yes
post	/launch-openaccount/services/contact-handler/prospect-product-selection/b...	0	1	Yes
get	/launch-openaccount/services/session-dialogue/onboarding-journey/registe...	0	1	Yes

# Vulnerabilities

3 vulnerability findings detected

/launch-openaccount/services/party-data-management/party-profile (1 finding)

incremental **MEDIUM (CVSS: 5.0)**

Category: idor  
Method: post /launch-openaccount/services/party-data-management/party-profile  
Last Detected: December 12, 2025 at 08:53 PM

Vulnerable Parameter:

**uuid (Body)**  
2c706b0c-2c6d-4943-a16f-2372080f366e

**uuid (Response)**  
2c706b0c-2c6d-4943-a16f-2372080f366e

**requestId (Response)**  
13

Auth: Authorization  
OWASP:  
API1:2023  
Impact:  
May allow users to access other users' data by manipulating object identifiers

Remediation:

- Solution**
1. Enforce object-level authorization on every request (never rely on client-side checks).
  2. Validate the requested object belongs to the authenticated user/tenant before returning data.
  3. Use unguessable identifiers (UUIDs) only as defense-in-depth (still require authorization).
  4. Add tests for cross-user access and alert on repeated authorization failures.

- Exploitation**
- An attacker changes an object identifier (path/body/query) to access another user's records.
  - Predictable/incremental IDs make enumeration and scraping significantly easier.

- References**
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>
  - OWASP Cheat Sheet: Authorization: [https://cheatsheetseries.owasp.org/cheatsheets/Authorization\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html)

**Description:**

This test looks for object identifiers that are incremental IDs. An incremental identifier is an identifier that increments in value in a predictable way. Currently, the test does this by checking if the identifier is numerical.

The test analyzes response payloads and searches them for fields that look like IDs. The heuristic used for this analysis looks for fields with the following patterns:

- 1) The field is `id` (case-insensitive)
- 2) The field ends with `id` (case-insensitive)

The second heuristic is prone to yield false positives in some cases, so please review carefully the detections raised by this test and flag any false positives. An incremental ID is a target for data scraping or unauthorized access attempts because they are easily guessed.

## REQUEST

post /launch-openaccount/services/party-data-management/party-profile

```
post https://blue.www.olb-gal0.dev.bmo.com/launch-openaccount/services/party-data-management/party-profile
x-request-id: 13
Authorization: Bearer eyJhbGciOiJIU201eSI6IjBMTI4Q0JDLUhmTjU2Iiwia2lkIjoieYmVjTzd5ZlB5RkFvcldiekhUSVbmbzBCd3NraGpDeU42T2VpemRrNldvQSI6ImN0eSI6IkpXVCJ9.W19lYUht0WVC6ylfTPqT2u1RwC2MitWGFsnqV_tX8ZXAce3we-YGRqCC9w0TbOL6hTInJrBLV1VliAYd8sGkl_a8ZWielwCY_5oA9LfjO_m8YGW2MJ7oQKLG9d_UrZqLnU9YcTzrpi9Kfd3HD1dimgjXQLhfa3thSWoA749g4o-CJZWst670dFeltzZq41itSTGK5EkCVP1NQzDTFV3yhIBIpJInUP0kKQqL2fNN9gy4ox7HskU4XkzYGD6zextNoNx2KsYZJleK_JLcnoYEAcllmXhZz2OdHP0n9WfD5oPompK9OMy9e4OVZkt3_nJ2f71ggJUWIEPvSSaSyukNw.5C35SkGatdOXwgvEfd3i0A.DpiNcEcleb2ah7mDfj8mhWs4QHsh6eKqp8wvwsOzQ9USM1Yw97N01WNaJnEwWxcC_R9UUtthudNHZVupN19J0FBjD_f2mpEgKrKnHDTYiFmHWMqg-F2sbeadPQ2qn3RmrjBb300ah2LgW0p8V2uGWCuiRar-zgVt6sSQK9QcTbYrvhtjzMPFNzDJxvQXY76oj7NYRlWM_KUTGm6EetDoyWQ2eyJtYFeq_9qX25KCOCOTrpUJkpuYw2sr7WD90A0RV1QSteyDkRxyEQ-eHEv11XUKABJWGzY4x93LNan-3d6HXC2v7zyDk4WPZzNg-wTlyn_TUETpg22zIHv72r803nU4hFDW08JSHQh8j5oUJ83bxtnvfaHtxwZOB-F6dosgtIXBTgVn9B6t5hVLzwlRAIgFd-aPoD2UqrUo3Rppc8cz-Os2fo7-2rQ1Sz7cWXo5FRO_Cau3HpXw04nsGK9z6s55gefLkCz4uUFTVI6157GMxMidhTio8APIqfjQpfaH2X5rzJEUlOqwe3xUtK47Jwf4GvULHVItYjAlNcIhINEv4OonAYvk7xVvSnu7HJ5RnExWBjhHiETFluTiiuzC4TWRLA_t5vqdd6d-JE4M5Unbje6nHr1RZ28eVDF62CetxVOMy8pgAJJKePsBeVAvscj7kgf-hX3fyFqriFS-Gzjc8TliG2hzSEqV-Teii-SBybKuT7wMASostQoRgFpt8IallyvTxAFiZoBleuhk_BDOtrYdFIDVALElVIOtqn-hT36uOscKuvB5_P2lZyxFMZDcufzkWpcVmyps4pb5gHXaz037rYWFZrctltg29Jbo62gXf6DCOX-GarEUUwGosXOnBiM2HulhjtLV5-sleY4nk0YndFDTwzZuhBag3mCoKJHTGj8G_4opArvOCeZskBOEwU08iGp_x_PS8xyHNH1Mu8RVvnCKejZ8MAukNMKOkd2CEKa2MCSAJ2XRA4zdbxOsyCb7xDaj77fBy5ifaF9v27pvgobDgNE25PbHyyKS-4A00Hjhl_wuv3uNO-pqy6xmNEYZNHMDqFqOrmhjtU98xcFtGZ0Nd4ZJHsv29cobm_UIeNnDPxxQDMPguWDVoJ2UJMvakIs-4evK5_7wlwrKcFY402xm78i1IEgXpjy03qGb6WH3bnsZ-Nxt6PuhpsR9MO7JA2gbXewIXjDYBX7ffn2Uqs_fDWXh0g2zGJ4Jc8G4KjAal9iHL9J5B0PEUQa0_9qdzMRGZfa_pfuhpFYJUd3go0WjJi4UZKVUMPsSrJCsr1j4Uhg5Mk6hRCKx8MF60cnIMAE_i992KZgdubkN8ZoIKGxRJLPjYskly6Udni2gMgAmAgN1b39oj-HthB_3YCNgap4GNY1FlD5_AaPF8Mua9oFd0TGNF8iTry3Q0ZhKHNGWDCSw7SuddIo5gieCcrUjx372lYrRF30F6UOQFAVa9A-n0aUfLqW6Bz2eI7TV7JdaJn4PxJ7KRnjXlaNKD25W6yvp_C-Z5kkMIw3JTA3Czcra9q3SDZr1cOCQRd10_p01fhJ8S-2bt7ZrNyK5uafHlnMgBsyV_6Fh07pjOMwkuRU9_Eu7EQ3D0xXK4E5xR9D9mBNLEUzZMqBrtw8YjOacJ196LL9WVFJtjr9ft24v1C4RxxwlUwlpj_DvNDz19KEXfp7sDOPMICADMGMCjtNbgD7Itghy0MTmKQeWu920ECOHpA96TswMb90cCtPkHqaxniJhNVWLTeMpy8htrA6CrLCEy348IofyXTuy-ZiueGJgcsfYZNTSjVq7bDSWMAU1J3M5njmmU2QOCR2j5_w8Q413M7SU3-tPqTrB1_kZIGJtMoQJ6ir2eNjIePilXi68QKB8VWIMzBHSYm16P8EpaRHSOuhRvizI_UalaiFAYKRL-HkZZVnI05arsZWqThLmZd7EYlEQcEbKhTCNhnJkiAiBu_4giqOeVsVhDP5cnom0kKtNoJgwvfq3qcmhEnUIDC3udNMGRKjwGBTk9HZL1VquyfSez99aq-Wv3-Y9Rxs8GPN5ExGZwD6Jvt_IP8IbigoEptm0p4qrck-XtbgaIV6VIS6yap9Z3G8zSJXUz2Og66NZWapa-mZfbIjwEoBOSczWINxUYNWxIj50xxOf0A_iwLRMu7dvlxkxBLWQQX1j03DdoYfmjwmZ6paIjFwVsZAcpcYhdx8bWVCC93EtX0uSgeG3yCIvFAlPHkvsPRZ3NfLgSVwRjyQzIkkWWXESkFfKCCX0C1Af0tRqWIMQ75toFi664WcN9MfaFpeox7WnJ_TJfgyEK1-ar7siaD1skc7IAaEPndusw6WR2mW5AxOvpFFHceLpQYEDHonaiZTTeJ1lhmromU1lheiUlKAc9C2JUysGZjJB-SHuz8k_gNaw0yTt1EXdXf6hnb9Ia3FePgJM0ty6Dc37WdDm_8LXjNF8FMzJWS0901IdA0iRbd9snQ_sG6kvyyddTl0_JdrhDAkWx1lekvnJQkAgqoZ7ucgUondeJw-BwLtrKv0sb0Eik7YYES6b4pV6uu5Xach0HrevPlhsOX1DICQIE6USg4cWcdKFBAewn7qZ8WIkVlJxGk8P2Z8tWESqbbf548yOmZWPRDY1VeuMwYbOn98W0F08-Pgsy8MQ3pE0mG8RLH5WUSBASqB1DQNXOj5SKRSElYVf08LcA5MxJN2eUxxwXjNU3JSP0Kjklh4UAHMqYt1gml_9_dhCm2YwZnfIvyygZ-3VDjpC-QNpYe5x88iQNNziB8bpKP6bSTMtVfrAP7T8iPnIL-hB-T54AaR3jpXDeenpCQ0RLlXywgHibnVvS-xUtsPw_GxbF4ghRLWIGj3pYo9EN8wXDLiHiovvULGa9ONY25iAEQcFVV8pD4-W2g95Ehy2joJyid0CTtpTyZAp1UdX49E9e1VZ3cd7UN0JfFWvzrLenUM3z7VGknaywCl_m2EEBWhRCqg-_HBGBb7wtz17WFOfg.gdv7lNNdbHz_r2PE0u0lSw
x-fapi-interaction-id: 001
x-client-id: 001
x-api-key: 81f64d142eec79f3c669faa52023ef87-pr1
content-type: application/json
x-apigw-api-id: md8iaxf536

{"journeyEvent":{"eventId":"E2"},"originatorData":{"channel":"MOB"},"personalInfo":{"birthdate":"1999-01-01","firstName":"TestMockKYCPassw","lastName":"TestSurnamedws"},"uuid":"2c706b0c-2c6d-4943-a16f-2372080f366e"}
```

## RESPONSE

200

```
HTTP 200
x-request-id: 13
x-global-transaction-id: d1f1e4cd693c82210261f070
date: Fri, 12 Dec 2025 20:59:14 GMT
server: Server
x-amz-apigw-id: Vf1FUHGU4osF4yQ=
x-amzn-requestid: 44f092de-3101-4fda-b32e-bea0ba06a9ed
vary: Accept-Encoding
x-ratelimit-limit: name=default,800;
x-ratelimit-remaining: name=default,746;
x-amzn-trace-id: Root=1-693c8222-755deaf098dbdda1924dfcf
connection: keep-alive
content-type: application/json; charset=utf-8
akamai-grn: 0.74643017.1765573153.c27af73

{"status":"Success","uuid":"2c706b0c-2c6d-4943-a16f-2372080f366e","message":"No record is found","requestId":"13"}
```

## secure

MEDIUM (CVSS: 5.0)

**Category:** cookies  
**Method:** get /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:** API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your cookies are correctly configured. We examine the `Set-Cookie` header in the API responses, if present, and check if it's correctly configured.

It's generally not recommended to use cookies with APIs for credentials and sensitive data, as there are better alternatives to handle authentication, such as JSON Web Tokens.

Incorrectly configured cookies make your API vulnerable to credentials leakage via cross-site scripting (XSS) and other form of attacks.

Secure cookie configuration requires an expiration date for the cookie, after which the browser deletes its associated data. We set the cookie's expiration date using one of the following attributes:

- **\*\*Secure\*\***: it forces the browser to send cookies over HTTPS. Missing this attribute means exposing credentials over non-encrypted traffic. Hence, the absence of this attribute causes the test to raise a detection with CVSS 7.
- **\*\*HttpOnly\*\***: this attribute prevents cookies from being accessed or modified by JavaScript code. Missing this attribute means malicious could seize the cookie and use it for malicious purposes. Hence, missing this attribute raises a detection with CVSS 7.
- **\*\*Expires\*\***: the expiration date of the cookie.
- **\*\*Max-Age\*\***: the time, in seconds, for which the browser must keep the cookie data stores.

`Expires` and `Max-Age` control how long the cookies will be stored in the browser. Without these attributes, the cookies remain in the browser until the browser session closes completely. If your `Set-Cookies` header doesn't include one of these attributes, the test raises an informational detection with CVSS 7.

It is also best practice to restrict which domains can receive the cookie. You can constrain the domains using one of the following fields:

- **\*\*Domain\*\***: specifies the domain and associated subdomains that can receive the cookie. The browser will not send cookies to other domains.
- **\*\*SameSite\*\***: this attribute controls whether and how the browser forwards cookies to other sites. It has three options: 1) `Strict` only includes cookies in requests originating from your own website; 2) `Lax` includes cookies in redirects to other websites; 3) `None` includes cookies in all requests to other websites. By default, SameSite's value is Lax (SameSite=Lax), which means cookies are only forwarded on top-level redirects, e.g. when we click on a link to go to another site. A threat actor could exploit this feature if they find a way to insert a malicious link within your website. If this attribute is missing or has a permissive configuration, the test raises an information detection with CVSS 5.

To learn more about securely configuring cookies, check out the following resource: ["Using HTTP cookies"](https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies).

Failing Test Logs

REQUESTget /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/p...

get https://blue.www.olb-qa10.dev.bmo.com/launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection  
x-request-id: 13  
x-fapi-interaction-id: 001  
x-client-id: 001  
x-api-key: 81f64d142eec79f3c669faa52023ef87-pr1  
product\_id: 1  
content-type: application/json  
offer\_id: 1  
x-apigw-api-id: srgbjegrx2  
  
null

RESPONSE400

HTTP 400  
x-global-transaction-id: dl1f1e4cd693c8349026224c0  
date: Fri, 12 Dec 2025 21:04:09 GMT  
content-length: 134  
connection: keep-alive  
content-type: application/json  
akamai-grn: 0.56643717.1765573449.1a6f97c3  
  
{ "httpCode": "400", "httpMessage": "Bad Request", "moreInformation": "One or more required API parameters are missing in the API request." }

/launch-openaccount/services/session-dialogue/onboarding-journey/init (1 finding)

secureMEDIUM (CVSS: 5.0)

Category:cookies  
Method:get /launch-openaccount/services/session-dialogue/onboarding-journey/init  
Last Detected:December 12, 2025 at 08:53 PM  
OWASP:  
API8:2023  
  
Remediation:  
Solution  
1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.  
2. Avoid cookies for API credentials where possible; prefer token-based auth.  
3. Shorten session lifetime and rotate tokens/identifiers regularly.  
  
Exploitation  
• Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.  
  
References  
• MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>  
• OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>  
  
Description:  
This test checks whether your cookies are correctly configured. We examine the `Set-Cookie` header in the API responses, if present, and check if it's correctly configured.  
  
It's generally not recommended to use cookies with APIs for credentials and sensitive data, as there are better alternatives to handle

authentication, such as JSON Web Tokens.

Incorrectly configured cookies make your API vulnerable to credentials leakage via cross-site scripting (XSS) and other form of attacks.

Secure cookie configuration requires an expiration date for the cookie, after which the browser deletes its associated data. We set the cookie's expiration date using one of the following attributes:

- **\*\*Secure\*\***: it forces the browser to send cookies over HTTPS. Missing this attribute means exposing credentials over non-encrypted traffic. Hence, the absence of this attribute causes the test to raise a detection with CVSS 7.
- **\*\*HttpOnly\*\***: this attribute prevents cookies from being accessed or modified by JavaScript code. Missing this attribute means malicious could seize the cookie and use it for malicious purposes. Hence, missing this attribute raises a detection with CVSS 7.
- **\*\*Expires\*\***: the expiration date of the cookie.
- **\*\*Max-Age\*\***: the time, in seconds, for which the browser must keep the cookie data stores.

`Expires` and `Max-Age` control how long the cookies will be stored in the browser. Without these attributes, the cookies remain in the browser until the browser session closes completely. If your `Set-Cookies` header doesn't include one of these attributes, the test raises an informational detection with CVSS 7.

It is also best practice to restrict which domains can receive the cookie. You can constrain the domains using one of the following fields:

- **\*\*Domain\*\***: specifies the domain and associated subdomains that can receive the cookie. The browser will not send cookies to other domains.
- **\*\*SameSite\*\***: this attribute controls whether and how the browser forwards cookies to other sites. It has three options: 1) `Strict` only includes cookies in requests originating from your own website; 2) `Lax` includes cookies in redirects to other websites; 3) `None` includes cookies in all requests to other websites. By default, SameSite's value is Lax (SameSite=Lax), which means cookies are only forwarded on top-level redirects, e.g. when we click on a link to go to another site. A threat actor could exploit this feature if they find a way to insert a malicious link within your website. If this attribute is missing or has a permissive configuration, the test raises an information detection with CVSS 5.

To learn more about securely configuring cookies, check out the following resource: ["Using HTTP cookies"](https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies).

## Failing Test Logs

**REQUEST** get /launch-openaccount/services/session-dialogue/onboarding-journey/init

```
get https://blue.www.olb-qa10.dev.bmo.com/launch-openaccount/services/session-dialogue/onboarding-journey/init
x-request-id: 10
x-fapi-interaction-id: 001
x-client-id: 001
x-api-key: 81f64d142eec79f3c669faa52023ef87-pr1
content-type: application/json

null
```

**RESPONSE**

400

```
HTTP/1.1 400
x-global-transaction-id: d1f1e4cd693c818601b65873
date: Fri, 12 Dec 2025 20:56:38 GMT
content-length: 134
connection: keep-alive
content-type: application/json
akamai-grn: 0.74643017.1765572998.c1ea15f

{"httpCode": "400", "httpMessage": "Bad Request", "moreInformation": "One or more required API parameters are missing in the API request."}
```

# Informational Findings

40 informational findings detected

/launch-openaccount/services/party-data-management/party-profile (6 info findings)

hsts INFO

**Category:** headers  
**Method:** patch /launch-openaccount/services/party-data-management/party-profile  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

Remediation:

Solution

- 1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
- 2. Avoid cookies for API credentials where possible; prefer token-based auth.
- 3. Shorten session lifetime and rotate tokens/identifiers regularly.

Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

rateLimit INFO

**Category:** headers  
**Method:** patch /launch-openaccount/services/party-data-management/party-profile  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

Remediation:

Solution

- 1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
- 2. Avoid cookies for API credentials where possible; prefer token-based auth.
- 3. Shorten session lifetime and rotate tokens/identifiers regularly.

Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.  
It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The

RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

## hsts

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/party-data-management/party-profile  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/party-data-management/party-profile  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

## hsts

INFO

**Category:** headers  
**Method:** get /launch-openaccount/services/party-data-management/party-profile  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** get /launch-openaccount/services/party-data-management/party-profile  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

## References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

## Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/pa...

## hsts

### INFO

**Category:** headers  
**Method:** patch /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection  
**Last Detected:** December 12, 2025 at 08:53 PM

## OWASP:

API8:2023

## Remediation:

### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

## References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

## Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## network

### INFO

**Category:** infoLeak  
**Method:** patch /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection  
**Last Detected:** December 12, 2025 at 08:53 PM

## OWASP:

API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API response headers contain any network details that could potentially be revealing information about internal network topology. The test analyzes all the headers in the response looking for IPv4 and IPv6 patterns.

Revealing internal network configuration allows threat actors to gain insights about your network topology. Such information can be leveraged in SSRF attacks to target specific addresses within your network, or in a server breach to achieve lateral movement between networks.

## hsts

### INFO

**Category:** headers

**Method:** post /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection

**Last Detected:** December 12, 2025 at 08:53 PM

#### OWASP:

API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## hsts

### INFO

**Category:** headers

**Method:** get /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection

**Last Detected:** December 12, 2025 at 08:53 PM

#### OWASP:

API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers

**Method:** get /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection

**Last Detected:** December 12, 2025 at 08:53 PM

#### OWASP:

API8:2023

API4:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

## hsts

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/party-data-management/party-profile/address-compliance/get  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/party-data-management/party-profile/address-compliance/get  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
  - RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
  - RateLimit-Reset: specifies how much time is left in the current time window.
- You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

## network

### INFO

**Category:** infoLeak  
**Method:** post /launch-openaccount/services/party-data-management/party-profile/address-compliance/get  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:** API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

##### Description:

This test checks whether the API response headers contain any network details that could potentially be revealing information about internal network topology. The test analyzes all the headers in the response looking for IPv4 and IPv6 patterns.

Revealing internal network configuration allows threat actors to gain insights about your network topology. Such information can be leveraged in SSRF attacks to target specific addresses within your network, or in a server breach to achieve lateral movement between networks.

/launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/st...

## hsts

### INFO

**Category:** headers  
**Method:** patch /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/status  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:** API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

## References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

## Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers

**Method:** patch /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/status

**Last Detected:** December 12, 2025 at 08:53 PM

### OWASP:

API8:2023

API4:2023

## Remediation:

### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

## References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

## Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The

RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API.

Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/session-dialogue/onboarding-journey/status (2 info findings)

## hsts

INFO

**Category:** headers

**Method:** post /session-dialogue/onboarding-journey/status

**Last Detected:** December 12, 2025 at 08:53 PM

### OWASP:

API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** post /session-dialogue/onboarding-journey/status  
**Last Detected:** December 12, 2025 at 08:53 PM

#### OWASP:

API8:2023  
API4:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

## hsts

INFO

**Category:** headers

**Method:** post /launch-openaccount/services/session-dialogue/prospect-product-selection/bmo-web/get

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:**

API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## network

INFO

**Category:** infoLeak

**Method:** post /launch-openaccount/services/session-dialogue/prospect-product-selection/bmo-web/get

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:**

API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API response headers contain any network details that could potentially be revealing information about internal network topology. The test analyzes all the headers in the response looking for IPv4 and IPv6 patterns.

Revealing internal network configuration allows threat actors to gain insights about your network topology. Such information can be leveraged in SSRF attacks to target specific addresses within your network, or in a server breach to achieve lateral movement between networks.

## hsts

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/session-dialogue/onboarding-journey/registered-device  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/session-dialogue/onboarding-journey/registered-device  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
  - RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
  - RateLimit-Reset: specifies how much time is left in the current time window.
- You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/session-dialogue/onboarding-journey/delivery-target-v...

## hsts

INFO

**Category:** headers

**Method:** post /launch-openaccount/services/session-dialogue/onboarding-journey/delivery-target-verification

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers

**Method:** post /launch-openaccount/services/session-dialogue/onboarding-journey/delivery-target-verification

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/pa...

hsts INFO

**Category:** headers

**Method:** post /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection/validation

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:** API8:2023

Remediation:

- Solution**
1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
  2. Avoid cookies for API credentials where possible; prefer token-based auth.
  3. Shorten session lifetime and rotate tokens/identifiers regularly.

**Exploitation**

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

- References**
- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
  - OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

**Description:**

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

network INFO

**Category:** infoLeak

**Method:** post /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/party-product-selection/validation

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:** API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API response headers contain any network details that could potentially be revealing information about internal network topology. The test analyzes all the headers in the response looking for IPv4 and IPv6 patterns.

Revealing internal network configuration allows threat actors to gain insights about your network topology. Such information can be leveraged in SSRF attacks to target specific addresses within your network, or in a server breach to achieve lateral movement between networks.

/launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/do...

## hsts

### INFO

**Category:** headers

**Method:** post /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/documents/get

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:**  
API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/product-sales-agreement/everyday-banking-agreement/documents/get  
**Last Detected:** December 12, 2025 at 08:53 PM

### OWASP:

API8:2023

API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/processing-order/id-verification-and-credit-risk-adju...

## hsts

INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/processing-order/id-verification-and-credit-risk-adjudication  
**Last Detected:** December 12, 2025 at 08:53 PM

### OWASP:

API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

## References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

## Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers

**Method:** post /launch-openaccount/services/processing-order/id-verification-and-credit-risk-adjudication

**Last Detected:** December 12, 2025 at 08:53 PM

### OWASP:

API8:2023

API4:2023

## Remediation:

### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

## References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

## Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/processing-order/kyc-evaluation-and-credit-risk-adjud...

## hsts

INFO

**Category:** headers

**Method:** post /launch-openaccount/services/processing-order/kyc-evaluation-and-credit-risk-adjudication

**Last Detected:** December 12, 2025 at 08:53 PM

### OWASP:

API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers

**Method:** post /launch-openaccount/services/processing-order/kyc-evaluation-and-credit-risk-adjudication

**Last Detected:** December 12, 2025 at 08:53 PM

#### OWASP:

API8:2023

API4:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

## hsts

INFO

**Category:** headers  
**Method:** get /session-dialogue/onboarding-journey/end  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** get /session-dialogue/onboarding-journey/end  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
  - RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
  - RateLimit-Reset: specifies how much time is left in the current time window.
- You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/session-dialogue/onboarding-journey/status (2 info fi...

## hsts

INFO

**Category:** headers  
**Method:** get /launch-openaccount/services/session-dialogue/onboarding-journey/status  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## rateLimit

INFO

**Category:** headers  
**Method:** get /launch-openaccount/services/session-dialogue/onboarding-journey/status  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023  
API4:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/session-dialogue/onboarding-journey/init (2 info find...

hsts INFO

Category: headers

Method: get /launch-openaccount/services/session-dialogue/onboarding-journey/init

Last Detected: December 12, 2025 at 08:53 PM

OWASP:

API8:2023

Remediation:

- Solution**
1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
  2. Avoid cookies for API credentials where possible; prefer token-based auth.
  3. Shorten session lifetime and rotate tokens/identifiers regularly.

- Exploitation**
- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

- References**
- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
  - OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

**Description:**

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

rateLimit INFO

Category: headers

Method: get /launch-openaccount/services/session-dialogue/onboarding-journey/init

Last Detected: December 12, 2025 at 08:53 PM

OWASP:

API8:2023

API4:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether the API includes the RateLimit header in the responses. If responses don't contain a rate limiting header, the test raises an informational detection.

It's best practice to implement rate-limiting policies for APIs, and to document those policies using the RateLimit header. The RateLimit header makes it clear for API consumers how often they can call the API, hence avoiding accidental misuse of the API. Rate limiting should be implemented on a per-endpoint or per-user-flow basis, with more sensitive endpoints having stricter rate-limit policies.

You can document your rate-limiting policy with the following headers:

- RateLimit-Limit: specifies how many requests are allowed for a given window of time.
- RateLimit-Remaining: specifies how many requests are left to the user in the current time window.
- RateLimit-Reset: specifies how much time is left in the current time window.

You can learn more about the RateLimit header with the 'RateLimit Header Fields for HTTP' document, by R. Polli and A. Martinez (<https://www.ietf.org/archive/id/draft-polli-ratelimit-headers-02.html>).

/launch-openaccount/services/contact-handler/prospect-product-selection/bmo-web (1...

## hsts

### INFO

**Category:** headers  
**Method:** post /launch-openaccount/services/contact-handler/prospect-product-selection/bmo-web  
**Last Detected:** December 12, 2025 at 08:53 PM  
**OWASP:**  
API8:2023

#### Remediation:

##### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

##### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

##### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.

## hsts

INFO

**Category:** headers

**Method:** get /launch-openaccount/services/session-dialogue/onboarding-journey/registered-device/2c706b0c-2c6d-4943-a16f-2372080f3

**Last Detected:** December 12, 2025 at 08:53 PM

**OWASP:**

API8:2023

### Remediation:

#### Solution

1. Set cookies with Secure, HttpOnly, and SameSite attributes appropriately.
2. Avoid cookies for API credentials where possible; prefer token-based auth.
3. Shorten session lifetime and rotate tokens/identifiers regularly.

#### Exploitation

- Missing cookie attributes can enable session theft via XSS, downgrade, or cross-site request scenarios.

#### References

- MDN: Set-Cookie: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- OWASP API Security Top 10: <https://owasp.org/www-project-api-security/>

#### Description:

This test checks whether your responses contain a header named HTTP 'Strict-Transport-Security' (HSTS) and whether it is correctly configured. Securely configured HSTS protect your users by forcing browsers to use HTTPS, hence mitigating the risk of data leaks in man-in-the-middle and other attacks.