securitum

Security report

SUBJECT

„[APPNAME]” web application

DATE

30.03.2026 – 13.04.2026

REPORT DELIVERY DATE

13.04.2026

LOCATION

Poznan (Poland)

AUTHORS

Grzegorz Bronka
Maksym Hensitskyi

VERSION

1.0

securitum

Executive summary

This document is a summary of work conducted by Securitum. The subject of the test was the [redacted] web application available at https://[redacted]/.

Tests were conducted using the following roles:

- Power user,
- User,
- Unauthenticated user (visitor of the website).

No severe vulnerabilities were identified during the assessment. A few low-risk vulnerabilities and information points were reported.

The testing of the web application was conducted based on a fixed effort of 17 man-days. Within this allocated timeframe, the assessment covered the agreed scope of the application, with sufficient depth to review the key functionalities and security mechanisms. The approach was aimed at providing a thorough evaluation of the most relevant security risks and identifying vulnerabilities across the defined scope.

During the tests, particular emphasis was placed on vulnerabilities that might in a negative way affect confidentiality, integrity or availability of processed data.

The security tests were carried out according to generally accepted methodologies, including: OWASP TOP10, (in a selected range) OWASP ASVS as well as internal good practices of conducting security tests developed by Securitum.

Both auditors conducting this assessment hold the Offensive Security Certified Professional (OSCP) certification:

- https://credentials.offsec.com/9f00309f-8e3b-45c4-9b21-2f346d2881c0#acc.vQoyGLzc
- https://credentials.offsec.com/1a1b245e-01af-47ca-a35a-e5a55df2e666#acc.f5m7JUVA

An approach based on manual tests (using the above-mentioned methodologies), supported by several automatic tools (i.a. Burp Suite Professional, dirsearch, feroxbuster, nmap, testssl), was used during the assessment.

The vulnerabilities are described in detail in further parts of the report.

Risk classification

Vulnerabilities are classified on a five-point scale, that reflects both the probability of exploitation of the vulnerability and the business risk of its exploitation. Below, there is a short description of the meaning of each of the severity levels:

- CRITICAL – exploitation of the vulnerability makes it possible to compromise the server or network device, or makes it possible to access (in read and/or write mode) data with a high degree of confidentiality and significance. The exploitation is usually straightforward, i.e. an attacker does not need to gain access to the systems that are difficult to reach and does not need to perform social engineering. Vulnerabilities marked as 'CRITICAL' must be fixed without delay, mainly if they occur in the production environment.

- HIGH – exploitation of the vulnerability makes it possible to access sensitive data (similar to the 'CRITICAL' level), however the prerequisites for the attack (e.g. possession of a user account in an internal system) make it slightly less likely. Alternatively, the vulnerability is easy to exploit, but the effects are somehow limited.

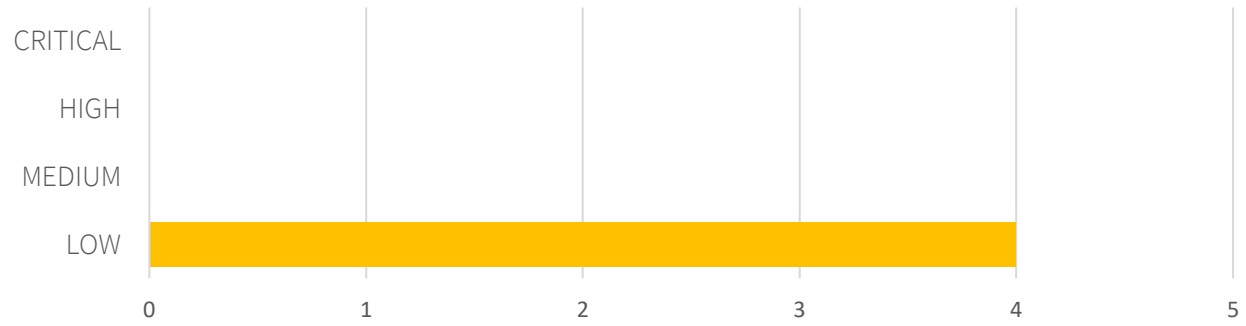
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing the user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser degree of significance.

- LOW – exploitation of the vulnerability results in minor direct impact on the security of the test subject or depends on conditions that are very difficult to achieve in practical manner (e.g. physical access to the server).

- INFO – issues marked as 'INFO' are not security vulnerabilities per se. They aim to point out good practices, the implementation of which will lead to the overall increase of the system security level. Alternatively, the issues point out some solutions in the system (e.g. from an architectural perspective) that might limit the negative effects of other vulnerabilities.

Statistical overview

Below, a statistical summary of vulnerabilities is shown:



Additionally, 5 INFO issues were reported.

Contents

Change history

Document date	Version	Change description
13.04.2026	1.0	Creation of the security report.

Vulnerabilities in the web application

[LOW] SECURITUM-2424509-001: Arbitrary URL Redirection via returnUrl parameter on login page

SUMMARY

During the tests it was observed that the application accepts any input provided for the `returnUrl` parameter during the login process. Currently this allows attacker to craft a link which, after login, will redirect the user to an arbitrary external domain.

Reflected cros-site scripting (XSS) through the `returnUrl` parameter is also possible, but it is blocked due to the restrictive Content Security Policy (CSP) header used by the application.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

PREREQUISITES FOR THE ATTACK

User must open the crafted link and login.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 - redirection

To confirm the vulnerability, paste following link into a browser:

```
https://[redacted]/login?returnUrl=https://securitum.com
```

Next, login as any user.

The application redirects the user to the provided URL. The redirection happens through the JavaScript code it cannot be shown as a redirection request.

The above link will have no effect if the user is already logged in. To force the redirection, the following link can be used to first logout the user (it will still work for users who are not logged in):

```
https://[redacted]/logout?expired=true&returnUrl=https://securitum.com
```

Example #2 – XSS

To confirm the vulnerability, paste following link into a browser:

```
https://[redacted]/login?returnUrl=javascript:alert(document.domain)
```

Next, login as any user.

The login attempt is interrupted by CSP:

If the CSP was disabled, the JavaScript would execute after login:

Similarly to the previous example, the user can be forced to logout through the link. The payload that works for both logged in and logged out users is:

```
https://[redacted]/logout?expired=true&returnUrl=javascript:alert(document.domain)
```

LOCATION

https://[redacted]/login?returnUrl=[payload]

RECOMMENDATION

It is recommended to verify the destination address to which the redirection takes place, e.g. by creating a list of allowed addresses to which users can be redirected (validation should take place on the server side).

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[LOW] SECURITUM-2424509-002: Redundant information revealed

SUMMARY

During the tests, it was observed that the application reveals detailed messages. An attacker using this fact may learn the application in more detail, including identification of the currently used software (e.g. the framework) and obtain valuable information that will help him or her to profile the application and plan further attacks.

PREREQUISITES FOR THE ATTACK

Access to the web application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 - The /api/auth endpoint

Below, there is an example of a request sent to the application:

```
GET /api/auth HTTP/2
Host: [redacted]
Cookie: [redacted]
Sec-Ch-Ua-Platform: [...]
Accept-Language: en
Accept: application/json, text/plain, */*
Sec-Ch-Ua: [redacted]
User-Agent: [redacted]
Sec-Ch-Ua-Mobile: ?0
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[redacted]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive
```

In response, the application reveals a detailed environment information:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Date: Sun, 05 Apr 2026 17:13:47 GMT
Cache-Control: no-store,no-cache
Content-Language: en
Pragma: no-cache
Set-Cookie:[redacted]
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000
X-Rate-Limit-Limit: 10s
X-Rate-Limit-Remaining: 9
X-Rate-Limit-Reset: 2026-04-05T17:13:57.7707062Z
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; frame-ancestors 'none'
Referrer-Policy: no-referrer
```

```
Permissions-Policy: accelerometer=(), autoplay=(), camera=(), display-capture=(), encrypted-media=(), fullscreen=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(), payment=(), picture-in-picture=(), publickey-credentials-get=(), screen-wake-lock=(), sync-xhr=(), usb=(), web-share=(), xr-spatial-tracking=()
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Resource-Policy: same-site

{
    "isFirstLogin": false,
    "passwordNeverExpires": false,
    "extendedInfo":
"{\"Function\":\"test2\",\"ImageUrl\":\"/media/images/users/[redacted].png?v=eXI_NnHZL9eYlY8RkFMWgRJM7pGsNuyeeVppqZZoUQw\"}",
    "roles": ["05c8249a-c2fe-4a0f-8dc3-b41101121993"],
    "permissions": ["Explore.Core.Automation.List", "Explore.Core.Automation.ListAny",
"Explore.Core.Automation.View", "Explore.Core.Automation.ViewAny",
"Explore.Core.Certification.CreateSuggestion", "Explore.Core.Company.AssignMemberAccountManager",
"Explore.Core.Company.AssignMemberAccountManagerAny",
"Explore.Core.Company.AssignMemberAccountManagerWithAutoAssign",
[…]
"Explore.Core.Employee.CreateEmployeeCertification",
"Explore.Core.Employee.CreateEmployeeCertificationAny",
"Explore.Core.Employee.CreateEmployeeEducation",
"Explore.Core.Employee.CreateEmployeeEducationAny",
"Explore.Core.Employee.CreateEmployeeProject", "Explore.Core.Employee.CreateEmployeeProjectAny",
"Explore.Core.Employee.CreateExternal", "Explore.Core.Employee.CreateInternal",
"Explore.Core.Employee.CreateNote", "Explore.Core.Employee.CreateNoteAny",
"Explore.Core.Employee.Delete", "Explore.Core.Employee.DeleteAny",
[…]
"Feniks.Storage.ViewresourcebookingAny", "Feniks.Storage.Viewresume",
"Feniks.Storage.ViewresumeAny", "Feniks.Storage.Viewtask", "Feniks.Storage.ViewtaskAny"],
    "id": "[redacted]",
    "userName": "[redacted]",
    "firstName": "Power User",
    "middleName": null,
    "lastName": "11+",
    "phoneNumber": null,
    "email": "[redacted]",
    "emailConfirmed": true,
    "status": "Active",
    "fullName": "Power User 11+"
}
```

Example #2 - /api/config/global

This endpoint returns extensive internal configuration details, including system-level GUIDs for users and roles (Anonymous, Authenticated, Admin and System), storage profile settings (allowed extensions and file sizes), and the Application Insights InstrumentationKey.

Knowledge of internal GUIDs could be used to craft more sophisticated IDOR or authorisation bypass attacks. The Instrumentation Key could potentially be used to inject malicious telemetry or retrieve logs from the Application Insights instance.

Request:

```
GET /api/config/global HTTP/2
Host: [redacted]
Cookie: [redacted]
Cache-Control: max-age=0
Sec-Ch-Ua: [...]
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: [...]
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: [redacted]
Accept: [redacted]
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Date: Sun, 05 Apr 2026 17:29:50 GMT
Cache-Control: no-store,no-cache
Content-Language: en
Pragma: no-cache
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; frame-ancestors 'none'
Referrer-Policy: no-referrer
Permissions-Policy: accelerometer=(), autoplay=(), camera=(), display-capture=(), encrypted-
media=(), fullscreen=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(),
payment=(), picture-in-picture=(), publickey-credentials-get=(), screen-wake-lock=(), sync-xhr=(),
usb=(), web-share=(), xr-spatial-tracking=()
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Resource-Policy: same-site


{
    "features": {
        "aiFeature": {
            "isEnabled": false
        },
        "cvBuilderFeature": {
            "isEnabled": true
        },
        "devFeature": {
            "isEnabled": false
        },
        "privacyFeature": {
```

```
            "isEnabled": false
        },
        "employeeFeature": {
            "isEnabled": true
[…]
        },
        "userRegistrationFeature": {
            "isEnabled": true
        },
        "workforceManagementFeature": {
            "isEnabled": false
        },
        "reportingFeature": {
            "isEnabled": true
        }
    },
    "feedback": {
        "receiverDisplayName": "[redacted]",
        "feedbackDefinitions": [{
            "title": "Send your compliment or comment",
            "caption": "This helps improve the application."
        }, {
           "type": "Negative",
            "title": "Send feedback",
            "caption": "Let the development team know what can be improved."
}    },
    "security": {
        "users": {
            "system": "[redacted]",
            "status": ["Inactive", "Active"]
        },
        "roles": {
            "anonymous": "[redacted]",
            "authenticated": "[redacted]",
            "administrator": "[redacted]",
            "system": "[redacted]"
        },
        "cookieExpireTimeInMinutes": 60,
        "sessionIdleTimeoutInMinutes": 0,
        "passwordValidationRegEx": "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*\\W)\\S{8,}",
        "enableExternalProviderAutoLogin": false,
        "enablePasswordReset": true,
        "enableLocalLogin": true
    },
    "webSockets": {
        "useDefaultEndpoint": true,
        "defaultEndpointPath": "/ws/events"
    },
    "storage": {
        "profiles": {
            "company": {
                "allowedExtensions": ["txt", "pdf", "doc", "docx", "xls", "xlsx", "ppt", "pptx",
"zip"],
                "maxFileSize": 31457280.0
[…]
    "googleSettings": {
```

```
        "recaptchaSiteKey": "[redacted]"
    },
    "applicationInsights": {
        "connectionString": "[redacted]"
    }
}
```

Example #3 – non-existent paths

The HTML response from the application contains a large, embedded JSON object (window.__remixContext), which exposes internal configuration and environment details.

Request

```
GET /nonexistent-path HTTP/2
Host: [redacted]
Connection: close
```

Response:

```
HTTP/2 404 Not Found
Server: openresty/1.21.4.2
Date: Mon, 30 Mar 2026 19:41:10 GMT
Content-Type: text/html; charset=utf-8
Set-Cookie: [redacted]
X-Enable-Cache: 0
Vary: Accept-Encoding

[…]
"onlineType\",\"APIDOC\",\"onlineId\",348857,\"projectId\",\"teamId\",2480,\"branchId\",312536,\"
visitType\",\"customDomain\",\"subdirectory\",\"specialFileType\",\"notification\",[],\"footerBan
ner\",[],\"projectSetting\",{\"_343\":344,\"_230\":346,\"_367\":368,\"_379\":380,\"_381\":382,\"_
383\":384},\"auth\",{\"_5\":345},\"securityscheme\",{\"_347\":42,\"_348\":42,\"_349\":58,\"_350
[…]
```

Example #4 – Host header changed:

```
GET / HTTP/2
Host: [redacted]
Accept-Language: en
Accept: application/json, text/plain, */*
User-Agent: […]
Referer: https://[redacted]/login?returnUrl=test
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

Response:

```
HTTP/2 404 Not Found
Content-Type: text/html
Date: Tue, 07 Apr 2026 14:19:21 GMT
Content-Length: 2667

<!DOCTYPE html>
<html>
<head>
    <title>Microsoft Azure Web App - Error 404</title>
    <style type="text/css">
        html {
            height: 100%;
[…]
```

LOCATION

Example #1:

- https://[redacted]/api/auth

Example #2:

- https://[redacted]/api/config/global

Example #3:

- https://[redacted]/nonexistent-path

Example #4:

- https://[redacted]

[LOW] SECURITUM-2424509-003: Possible DoS through image resizing

SUMMARY

During the tests, it was observed that the application allows users to request a resized image. Additionally, this functionality does not require authentication. An attacker could exploit this to launch a DoS attack on the application by repeatedly requesting image resizing.

PREREQUISITES FOR THE ATTACK

None – an image can be found on the login page and attacker can find parameters through brute forcing.

TECHNICAL DETAILS (PROOF OF CONCEPT)

To confirm the vulnerability, first send request for a regular image that does not require authentication (e.g. the LinkedIn icon from the landing page):

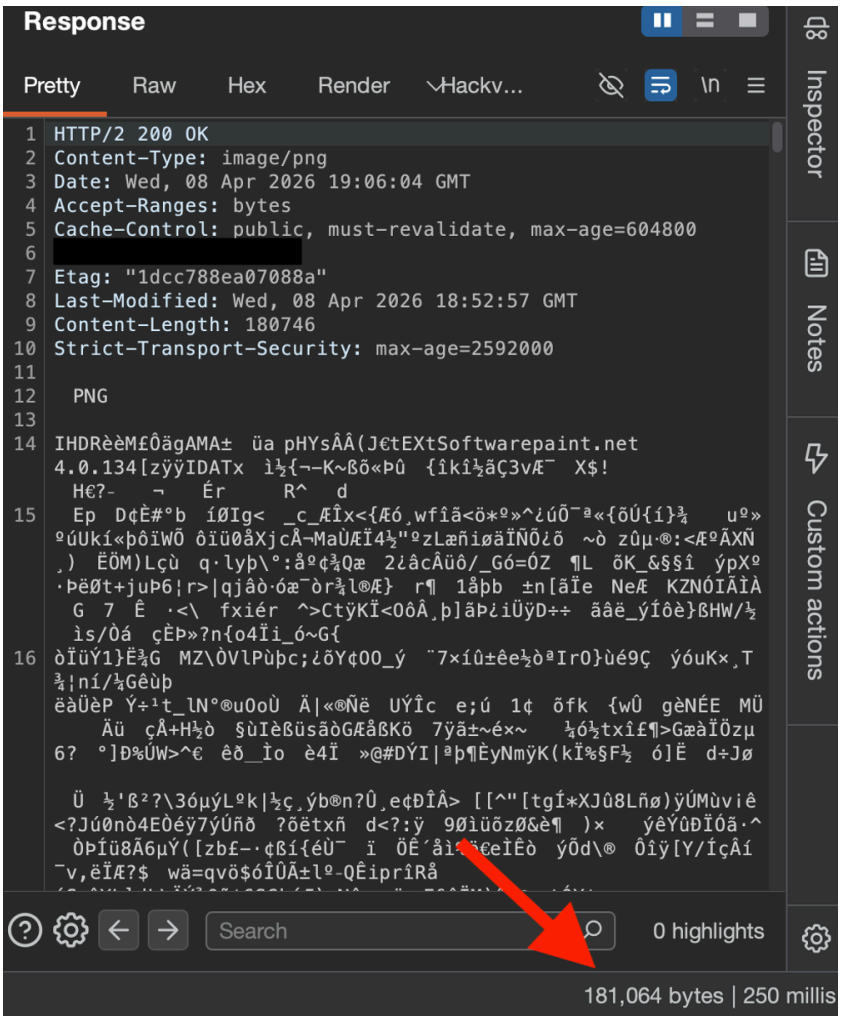
```
GET /images/login-providers/linkedin.png HTTP/2
Host: [redacted]
User-Agent: test
```

The response takes at maximum 200 milliseconds for the response and is only 939 bytes.

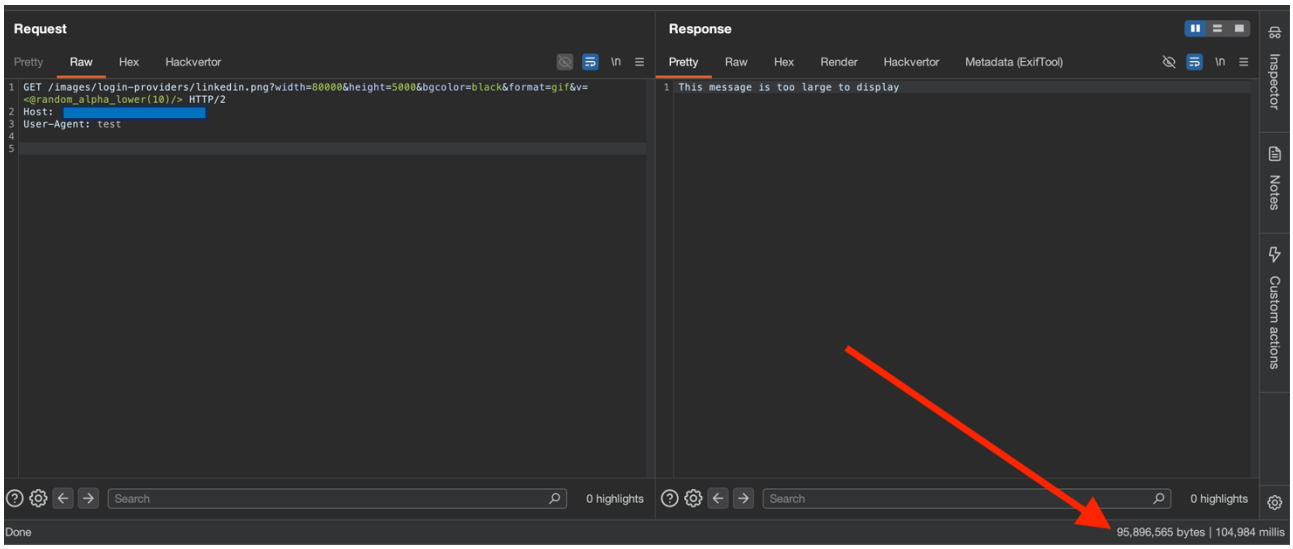
The image can be resized with either `height` or `width` parameters. For example, adding width 1000:

```
GET /images/login-providers/linkedin.png?width=1000 HTTP/2
Host: [redacted]
User-Agent: test
Content-Length: 1
```

The response is considerably larger:



By combining several parameters, including a random version of the image, the attacker can cause very large response:



When 10 requests were sent at the same time, some of them resulted in 502 responses signifying that the back-end server generating the images was possibly overwhelmed:

```
HTTP/2 502 Bad Gateway
Content-Type: text/html
Date: Thu, 09 Apr 2026 10:45:38 GMT
```

```
Content-Length: 1477

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<title>502 - Web server received an invalid response while acting as a gateway or proxy server.</title>
<style type="text/css">
[…]
```

Although the main application was still responsive this attack could potentially result in unnecessary resource usage, especially taking in consideration that it does not require authentication.

LOCATION

Vulnerability was observed in multiple places. Provided example would be easiest to find be an attacker.

RECOMMENDATION

It is recommended that all transformation parameters are restricted and validated - maximum values should be enforced for width and height, and only specific formats and quality should be allowed. It should be considered whether this functionality is necessary for all images – if it is not, it should be removed or limited to specific endpoints.

[LOW] SECURITUM-2424509-004: Software versions exposed

SUMMARY

During the audit, it was observed that the tested application returns redundant information in the HTTP response headers and body about the technologies in use. This behaviour can help an attacker to better profile the application environment, which then can be used to carry out further attacks.

More information:

- https://wiki.owasp.org/index.php/Testing_for_Web_Application_Fingerprint_(OWASP-IG-004)
- https://github.com/OWASP/OWASP-Testing-Guide/blob/master/4-Web-Application-Security-Testing/4.2.2%20Fingerprint%20Web%20Server%20(OTG-INFO-002)

PREREQUISITES FOR THE ATTACK

Access to the web application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1

Example of the HTTP request sent to the application:

```
GET /home HTTP/2
Host: [redacted]
Cookie: [redacted]
Sec-Ch-Ua: [...]
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: [...]
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: [Redacted]
Accept: [redacted]
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://[redacted]/login?returnUrl=%2Fhome
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

In response body, the application returns:

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Date: Sun, 05 Apr 2026 17:29:14 GMT
Cache-Control: no-store,no-cache
Content-Language: en
Pragma: no-cache
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: strict-origin-when-cross-origin
```

```
Content-Security-Policy: redacted
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: credentialless
Cross-Origin-Resource-Policy: same-site

<!DOCTYPE html>
<html>
<head>
    <title>[redacted]</title>
    <base href="/">
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable=0" />
    <meta name="description" content="">
    <meta name="author" content="">
    <meta name="version" content="4.10.0 (feniks 26.3.1-ci.44&#x2B;27a725b47f9bb5bda0cd3a7c128af51e1bbb69b5)">
    <meta name="apple-mobile-web-app-capable" content="yes">
[…]
```

Example #2

Request:

```
POST /[redacted]/api/v1/user-trackings HTTP/1.1
Host: [redacted]
Cookie: [redacted]
Content-Length: 224
Sec-Ch-Ua-Platform: [...]
X-Device-Id: tTq6ItCR-wvff-U3EW-uZzw-epBQLePUYCLh
Accept-Language: en-GB,en;q=0.9
Sec-Ch-Ua: [...]
Content-Type: application/json;charset=utf-8
Sec-Ch-Ua-Mobile: ?0
User-Agent: [redacted]
Accept: */*
Origin: https://[redacted]
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[redacted]/usage
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive

{"deviceId":"JZAQPenq-z7CB-Q4I7-RZ6j-aoWKWeLdE9fb","landingUrl":"https://[redacted]/usage","referrer":"http://burpsuite/","abTestingGroup":"","utmSource":"","utmMedium":"","utmCampaign":"","utmTerm":"","utmContent":""}
```

Response, containing detailed technologies in use in the Server header:

```
HTTP/2 200 OK
Server: openresty/1.21.4.2
Date: Tue, 31 Mar 2026 11:18:34 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 28
Vary: Accept-Encoding
Vary: Origin
Access-Control-Allow-Credentials: true

{"success":true,"data":null}
```

LOCATION

The issue affects entire application.

RECOMMENDATION

It is recommended to remove all unnecessary headers from the HTTP responses that reveal information about the technologies used.

Informational issues

[INFO] SECURITUM-2424509-005: No invalidation of the session after logout

SUMMARY

During the audit, it was found that the session is not invalidated when the “Logout” button is pressed. An attacker who successfully acquires the cookie will be able to use it to gain access to the application. Example.[redacted].App cookie will be able to use it to gain access to the application.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

TECHNICAL DETAILS (PROOF OF CONCEPT)

Analysis revealed that `POST /api/auth/logout` only empties the cookie locally. The server-side session remains active.

To confirm the vulnerability, the following steps were performed:

1. A user logged into the application.
2. The .Example.[redacted].App cookie was captured.
3. The user clicked the "Logout" button.
4. A request was sent using the invalidated cookie:

```
GET /api/auth HTTP/2
Host: [redacted]
Cookie: .Example.[redacted].App=CfD[…]Q1g
User-Agent: test
Accept: application/json, text/plain, */*
```

5. The application returned the following data, confirming the cookie remains active:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 10 Apr 2026 07:26:21 GMT
Cache-Control: no-store,no-cache
Content-Language: en
Pragma: no-cache
Set-Cookie: .Example.[redacted].Antiforgery=CfD[…]cQo; path=/; secure; samesite=strict; httponly
Set-Cookie: XSRF-TOKEN=CfD[…]8Cw; path=/; secure; samesite=none
Content-Length: 23910
Strict-Transport-Security: max-age=31536000
X-Frame-Options: DENY
X-Rate-Limit-Limit: 10s
X-Rate-Limit-Remaining: 9
X-Rate-Limit-Reset: 2026-04-10T07:26:31.9530985Z
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; frame-ancestors 'none'
Referrer-Policy: no-referrer
Permissions-Policy: accelerometer=(), autoplay=(), camera=(), display-capture=(), encrypted-
media=(), fullscreen=(), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(),
```

```
payment=(), picture-in-picture=(), publickey-credentials-get=(), screen-wake-lock=(), sync-
xhr=(), usb=(), web-share=(), xr-spatial-tracking=()
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Resource-Policy: same-site

{
  "isFirstLogin":false,
  "passwordNeverExpires":false,
  "extendedInfo":null,
  "roles":[
    "05c8249a-c2fe-4a0f-8dc3-b41101121993"
  ],
  "permissions":[
    "Explore.Core.Automation.Create",
    "Explore.Core.Automation.Delete",
[…]
  ],
  "id":"6c0[…]8b9",
  "userName":"[redacted]",
  "firstName":"Audy",
  "middleName":"",
  "lastName":"11++",
  "phoneNumber":null,
  "email":"[redacted]",
  "emailConfirmed":true,
  "status":"Active",
  "fullName":"Audy 11++"
}
```

LOCATION

https://[redacted]/api/auth - session management.

RECOMMENDATION

It is recommended to invalidate user's session immediately after the “Logout” button is pressed.

[INFO] SECURITUM-2424509-006: Lack of general field validation

SUMMARY

During the test, it was observed that most of the fields do not have the correctly implemented server-side verification. For example, special characters can be used in employee name and surname.

Vulnerability does not currently create a security risk, but if the application communicates with other systems, it may cause unexpected behaviour.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is an example of a request with fields marked in yellow that do not have proper server-side validation:

```
POST /api/employees HTTP/2
Host: [redacted]
Cookie: ai_user=Y+b[…]55Z; _gid=GA[…]41; .Example.[redacted].Antiforgery=CfD[…]xnw;
.Example.[redacted].App=CfD[…]Ieu; _ga_EEV1GT9DG9=GS2[…]$h0; _ga=GA[…]43; XSRF-TOKEN=CfD[…]PMQ;
ai_session=ctx[…]160
Content-Length: 253
X-Xsrf-Token: Cf[…]MQ
Request-Id: |8f[…]3
Traceparent: 00[…]3-01
User-Agent: […]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[redacted]
Referer: […]
[…]

{
  "firstName":"Pentestx`'\"><h1>%3ch1%3c{{8*8}}x\u003ch1\u003ex",
  "middleName":"Pentests<img src onerror=prompt>",
  "surname":"Surname<svg><animatetransform onbegin=prompt(1)>",
  "type":"External",
  "isFreelancer":true,
  "files":[
  ],
  "cvSkills":[
  ],
  "cvLanguages":[
  ]
}
```

In response, the application accepts the above request:

```
HTTP/2 201 Created
Content-Type: application/json; charset=utf-8
Date: Thu, 09 Apr 2026 16:59:55 GMT
Cache-Control: no-cache,no-store
Content-Language: en
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Location: https://[redacted]/api/employees/edb[…]c9c
Pragma: no-cache
```

```
Set-Cookie: .Example.[redacted].App=Cf[…]W5; path=/; secure; samesite=lax; httponly
Content-Length: 45
Strict-Transport-Security: max-age=31536000
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; frame-ancestors 'none'
Referrer-Policy: no-referrer
Permissions-Policy: […]
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Resource-Policy: same-site

{
  "id":"edb[…]c9c"
}
```

LOCATION

https://[redacted]/api/employees

RECOMMENDATION

It is recommended to validate all the data received from a user (rejection of values inconsistent with the template/format of a given field – whitelist approach), and then encode it on the output in relation to the context in which it is embedded (in all places of application, not only those specified in the description).

For this purpose, it should be verified whether the framework used by the application has built-in functions that implement the described recommendation.

It is worth noting that the server should not return the values that caused the error, but only indicate that: "*An incorrect value was entered in the XYZ field*", alternatively with additional information about allowed characters, e.g. "*Allowed characters are letters and numbers*".

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[INFO] SECURITUM-2424509-007: Inconsistent API Rate Limiting

SUMMARY

During the tests, it was found that the API has rate limiting but only on specific endpoints. An attacker exploiting this fact could potentially execute a Denial of Service (DoS) attack, disrupt logic, or cause other security consequences.

More information:

- https://owasp.org/API-Security/editions/2023/en/0xa4-unrestricted-resource-consumption/
- https://cwe.mitre.org/data/definitions/770.html
- https://owasp.org/www-community/attacks/Denial_of_Service

TECHNICAL DETAILS (PROOF OF CONCEPT)

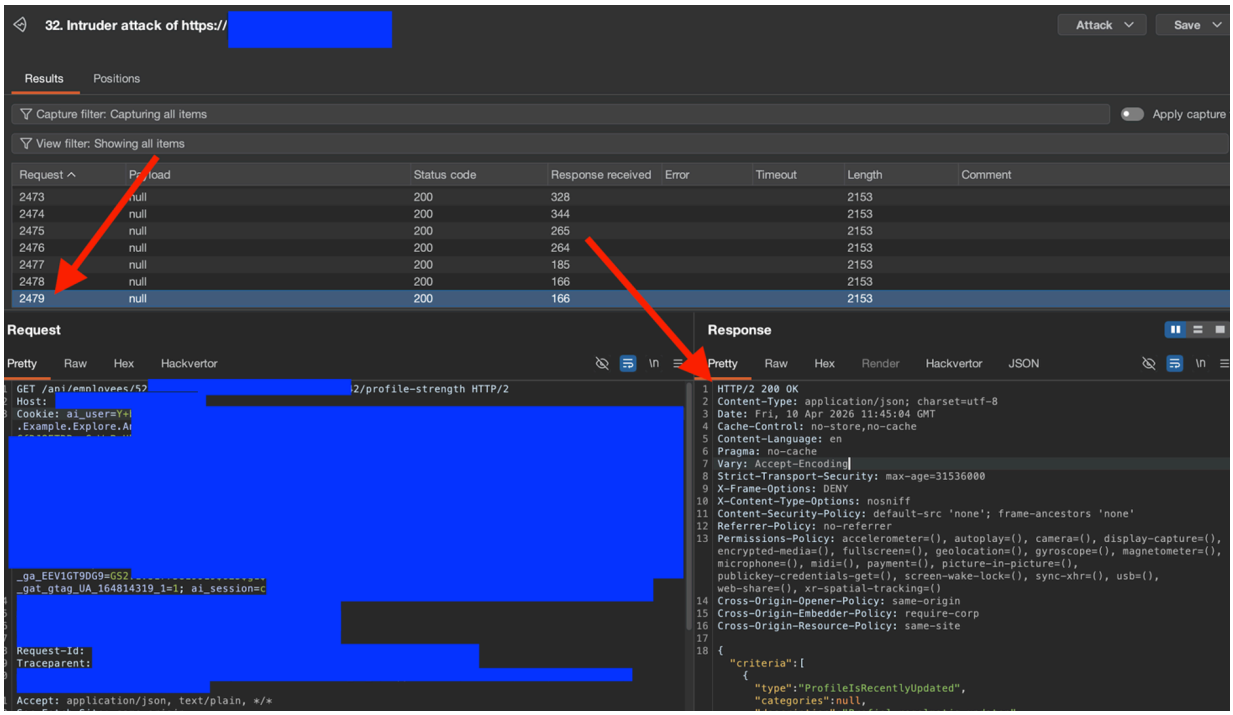
Specific endpoints such as https://[redacted]/api/auth have rate limiting - after more than 10 requests in 10 seconds the API starts to respond with:

```
HTTP/2 429 Too Many Requests
Content-Type: text/plain
Date: Fri, 10 Apr 2026 11:39:20 GMT
Cache-Control: no-cache, no-store
Content-Language: en
Pragma: no-cache
Retry-After: 3
Set-Cookie: XSRF-TOKEN=CfD[…]p7g; path=/; secure; samesite=none
Strict-Transport-Security: max-age=31536000
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'none'; frame-ancestors 'none'
Referrer-Policy: no-referrer
Permissions-Policy: […]
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Resource-Policy: same-site

API calls quota exceeded! maximum admitted 10 per 10s.
```

Similar limit should be implemented for all API endpoints.

The screenshot below shows over 2,000 requests sent to the `/api/employees/[employee id]/profile-strength` endpoint within a 10-second timeframe. All requests returned successful responses, proving no rate limiting is implemented on this endpoint:



LOCATION

https://[redacted]/api/employees/[employee id]/profile-strength

RECOMMENDATION

It is recommended to implement the rate limiting on all API endpoints.

[INFO] SECURITUM-2424509-008: Application accepts user supplied X-Real-Ip header

SUMMARY

The tested application accepts and processes user-supplied values submitted via the X-Real-Ip header.

This currently poses no security risk and is provided purely as informational finding for developers.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example request with the header and valid IP address:

```
GET / HTTP/2
Host: [redacted]
User-Agent: […]
Accept: application/json, text/plain, */*
X-Real-Ip: 127.0.0.1
```

In response, the server returns:

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: strict-origin-when-cross-origin
Content-Security-Policy: […]
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: credentialless
Cross-Origin-Resource-Policy: same-site
[…]

<!DOCTYPE html>
<html>
<head>
    <title>[redacted]</title>
    <base href="/">
    <meta charset="utf-8" />
[…]
```

Example request with invalid IP:

```
GET / HTTP/2
Host: [redacted]
User-Agent: […]
Accept: application/json, text/plain, */*
X-Real-Ip: 127.0.0.257
```

In response, the server returns error, proving the header is processed in some way:

```
HTTP/2 500 Internal Server Error
Date: Fri, 10 Apr 2026 12:13:00 GMT
Cache-Control: no-cache,no-store
Expires: -1
Pragma: no-cache
Content-Length: 0
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: strict-origin-when-cross-origin
Content-Security-Policy: […]
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: credentialless
Cross-Origin-Resource-Policy: same-site
```

LOCATION

https://[redacted]/

RECOMMENDATION

If this feature is not required by the application, it should be disabled.

[INFO] SECURITUM-2424509-009: Lack of Content-Disposition header

SUMMARY

From a security perspective, it is a good practice to add a `Content-Disposition` header to the HTTP API response, which will force the web browser not to interpret the response content under any circumstances. Forcing such behaviour on the application may protect the application against vulnerabilities, including for Cross-Site Scripting (XSS).

LOCATION

https://[redacted]/api/*

RECOMMENDATION

`Content-Disposition` header should be added in all server responses:

```
Content-Disposition: attachment; filename="api.json"
```

[INFO] SECURITUM-2424509-010: Lack of 2FA option

SUMMARY

During testing, it was found that there is no option to secure accounts using Two-Factor Authentication (2FA). 2FA is a mechanism that adds an extra layer of security; during the login process, after entering the correct credentials, the user is prompted to provide a Time-Based One-Time Password (TOTP)—a sequence of digits or characters generated by a device, such as a mobile app.

LOCATION

The web application authentication components.

RECOMMENDATION

It is recommended to implement 2FA as an optional feature that can be enabled at the user's request.

[INFO] SECURITUM-2424509-011: Support for parallel sessions

SUMMARY

The application supports parallel sessions on one user account. This allows one account to be used by several people at the same time (including when the session token is stolen). In the event that the current solution is retained, there should be a functionality that would be informing about active sessions, along with the option of disabling active user sessions.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#simultaneous-session-logons

PREREQUISITES FOR THE ATTACK

Theft of a session token or obtaining credentials.

LOCATION

Session management.

RECOMMENDATION

It is recommended to disable support for parallel sessions. It should be ensured that only one session can be active at the same account at the same time. An attempt to log into the account while another session is active should be confirmed, for e.g. with an SMS code or mobile or hardware token.

[INFO] SECURITUM-2424509-012: Suspicious user input transportation

SUMMARY

It was identified that the image upload endpoint (PUT `/api/users/current/image`) accepts malformed and potentially malicious payloads without proper server-side validation. Specifically, a crafted request containing a JavaScript URI scheme (javascript:alert('XSS')) prepended to a valid Base64-encoded PNG data URI was accepted by the server without rejection.

While no end-to-end exploit scenario leading to confirmed code execution was found — and the JavaScript payload was not triggered — the permissive backend validation logic represents a structural weakness that could facilitate an XSS attack if a complementary vulnerability were to be discovered in the future ((e.g., a reflected or stored rendering path). This issue has therefore been classified as INFO.

PREREQUISITES FOR THE ATTACK

- Authenticated access to the web application.
- Ability to issue PUT requests to `/api/users/current/image` on behalf of the authenticated user.

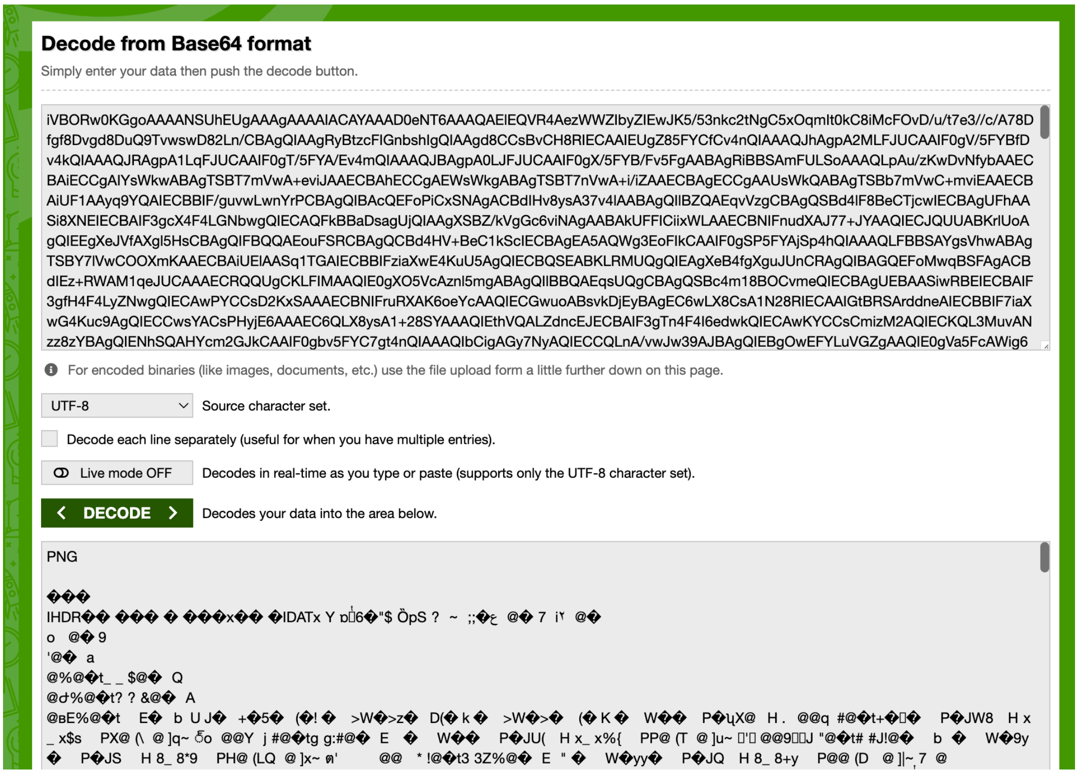
TECHNICAL DETAILS (PROOF OF CONCEPT)

An authenticated user can upload a profile image by sending a PUT request to `/api/users/current/image` with a JSON body containing the image as a Base64-encoded data URI (...). The following is an example of a legitimate request:

```
PUT /api/users/current/image HTTP/2
Host: [redacted]
Cookie: […]
Content-Length: 198258
Sec-Ch-Ua-Platform: [...]
X-Xsrf-Token: […]
Accept-Language: en
Sec-Ch-Ua: redacted
Sec-Ch-Ua-Mobile: ?0
Request-Id: [redacted]
Traceparent: [redacted]
User-Agent: […]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[redacted]
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[redacted]/mijn-account/settings
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"image":"
I3YG+FBjO0t8omYfz47nhRVP9NCh+eZmfOfrgT2K/WFXNbP0Ohvup1j+T+iH2Symbw7MhPmuE54X+jLP0gO1zmOL/aGHrvSMn
qNuBtrLh5od7KLPDDmIUZ0cZkzvnOrm/Nb0nKhN9KOz1NrDL83Zocjx2UlzZ2816lCXY7uDPQpNoSv0P0GvnqkVyxfvI9y/5v
hydiqdIa7vGaaDnvb3DO8tXE/7Wqu2Vn4i33O+s/SzdsZzJ/PUJ3aCGc3TFfkTuub8M3v+QMW6TjX3xGenP0yZPdE3455jT7i
P+mFG67U09WudaS4rV42c7y/n8b0Hdv3HP/5xsfal/YQe/TADzmFnTtw8e2Kv0pqeOeP/AAAA//8cPsH/AAAABklEQVQDAAfZ
VppV1t3gAAAAAElFTkSuQmCC"}
```

The following screenshot illustrates how the submitted value appears when decoded through an external Base64 analysis tool:



The following payload was crafted and submitted to the same endpoint by replacing the standard data URI scheme with a JavaScript URI prefix.The payload was structured so that the javascript: scheme appeared first, followed by a comment separator (//), and then a valid string — exploiting the server's substring-based content validation:

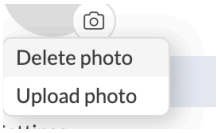
```
PUT /api/users/current/image HTTP/2
Host: [redacted]
Cookie: redacted
Content-Length: 159
Sec-Ch-Ua-Platform: [...]
X-Xsrf-Token: [redacted]
Accept-Language: en
Sec-Ch-Ua: redacted
Sec-Ch-Ua-Mobile: ?0
Request-Id: |[redacted]
Traceparent: [redacted]
User-Agent: [Redacted]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[redacted]
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[redacted]/mijn-account/basisgegevens
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

```
{
  "image":
"javascript:alert('XSS');//
C0lEQVR42mNkYAAAAAYAAjCB0C8AAAAASUVORK5CYII="
}
```

Fake PNG image structure:

```
iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAQAAAC1HAwCAAAAC0lEQVR42mNkYAAAAAYAAjCB0C8AAAAASUVORK5CYII=
```

The server returned a successful HTTP response, confirming that the malformed payload had been accepted and stored. The screenshot below shows the profile page after the upload, which confirms that the server did not reject the request:



The backend validation logic does not properly anchor its content-type check against the beginning of the submitted string. Instead of verifying that the payload strictly begins with (e.g., via a regex such as ^data:image\/png;base64,), the server appears to perform a substring search — checking whether the string exists anywhere within the input.

This allows an attacker to craft a value of the following structure:

```
javascript:alert('XSS');//<valid_base64_data>
```

In this payload, the javascript: scheme is the actual value of the field as the browser would interpret it; the substring satisfies the server's naive contains-check; and the // acts as a JavaScript comment, causing the remainder of the string to be ignored at execution time.

Although the JavaScript payload was not observed to execute in this test — due to the absence of a direct rendering path for the stored value — the structural flaw in the validation logic represents a genuine weakness that could be combined with other issues (e.g., a reflected or server-side rendered image src attribute) to create a stored or DOM-based XSS condition.

LOCATION

A [redacted] file upload API endpoint - PUT /api/users/current/image .

RECOMMENDATION

Validate all user-supplied data server-side using a strict whitelist approach. For the image upload field, ensure the value is validated against the full expected format, starting from the beginning of the string. After decoding the Base64 payload, verify that the actual binary content matches the expected image format (e.g., check the PNG/JPEG magic bytes). Do not rely solely on the MIME type declared in the data URI, as this is attacker-controlled. Ensure that any stored image references are properly HTML-encoded when rendered in the application, and that image src attributes do not accept JavaScript: URIs. Apply Content Security Policy (CSP) headers to restrict the execution of inline scripts. Verify whether the framework in use provides built-in mechanisms for file type and MIME type validation on upload endpoints, and enable these if they are available. Do not implement custom validation logic where a proven library is available.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html