Projet SERE

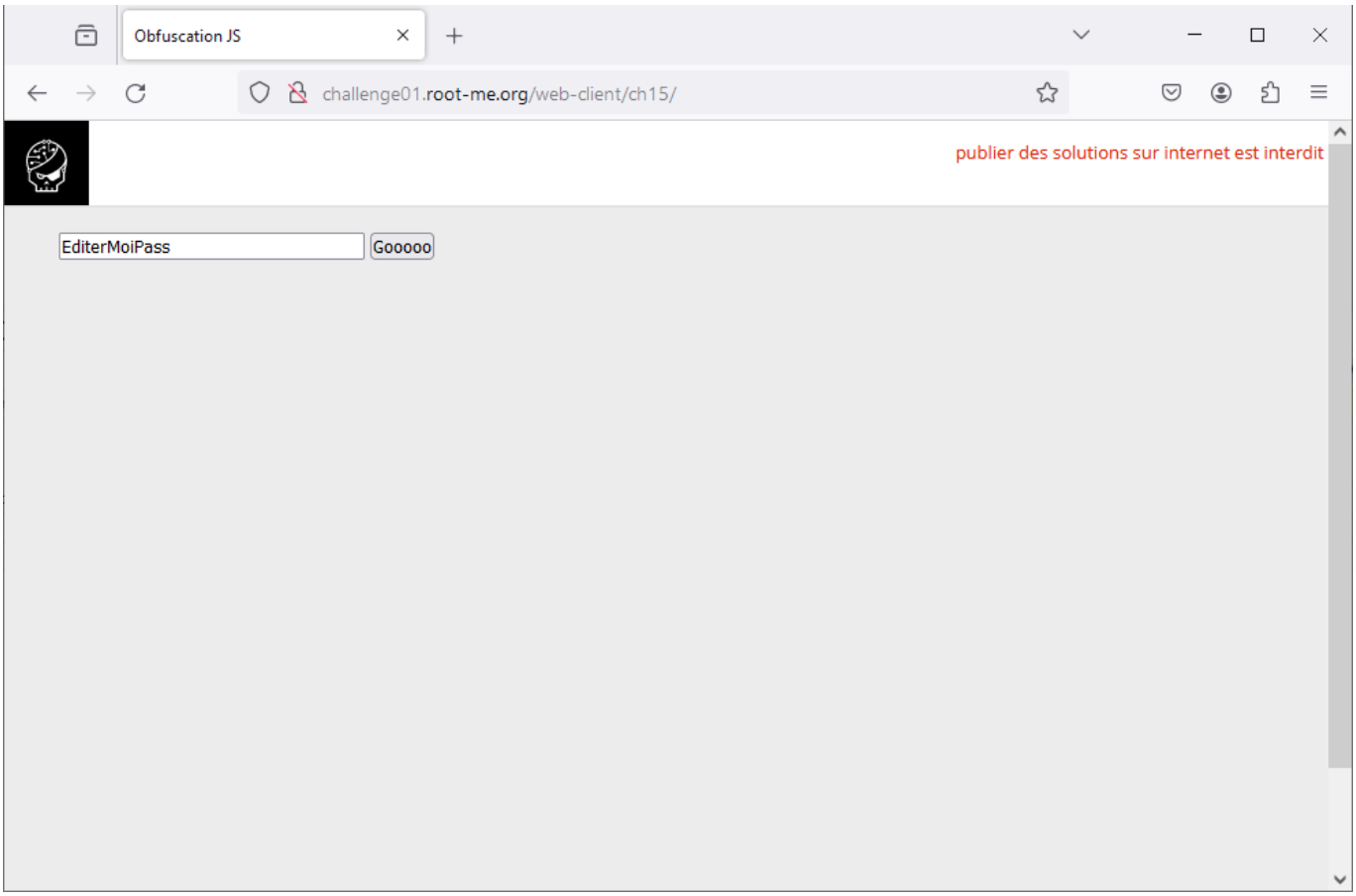
H4224 : Zijing WENG, Yikang SU, Remy ETIENNE, Le Tuan Khai NGUYEN, Mohamed-Ali LAJNEF

Javascript - Obfuscation 5 [Web-Clinet][70 Points]

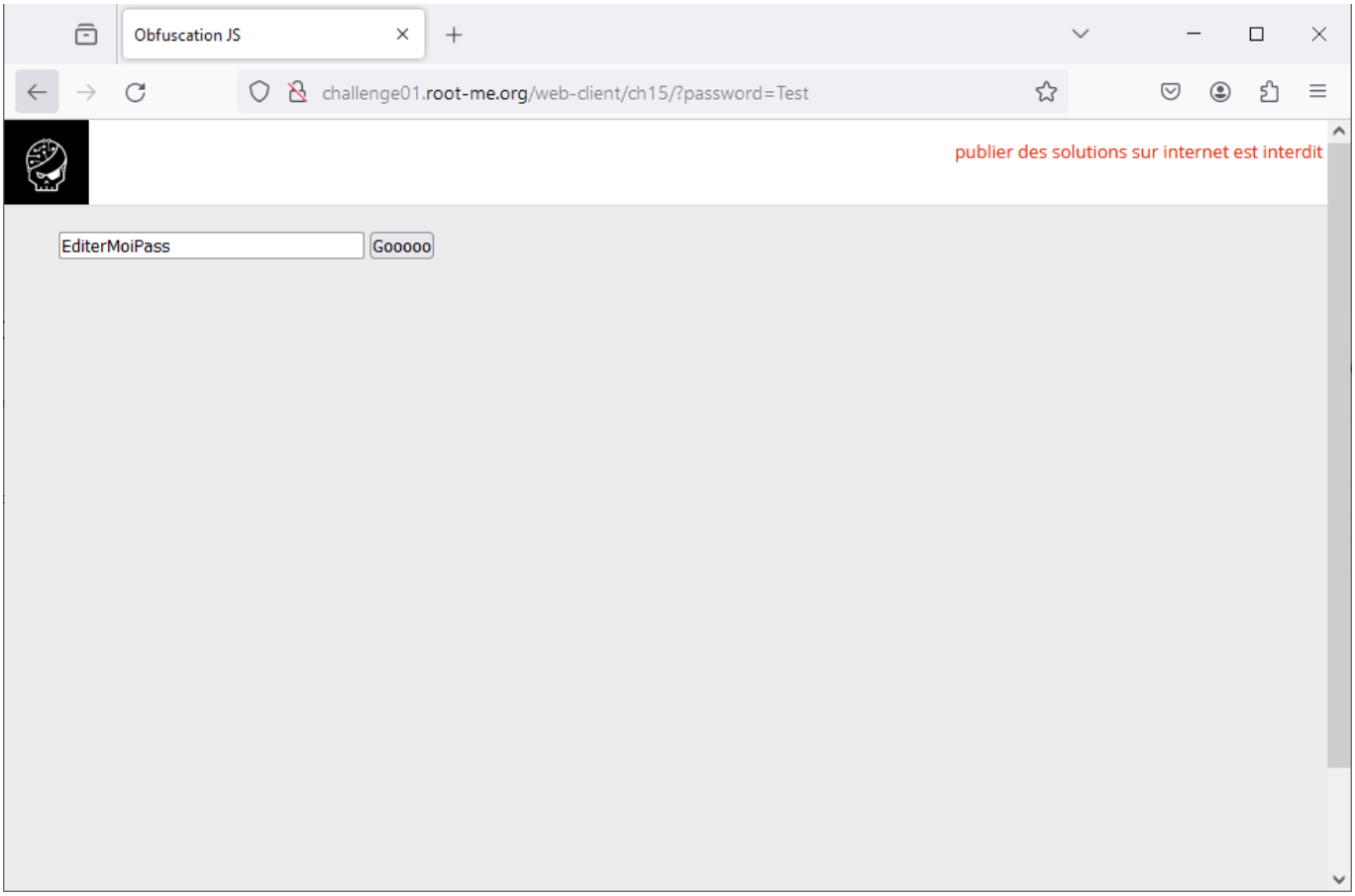
The problem can be found here: https://www.root-me.org/fr/Challenges/Web-Client/Javascript-Obfuscation-5
We have to understand a highly obfuscated Javascript code and find the password.

First steps

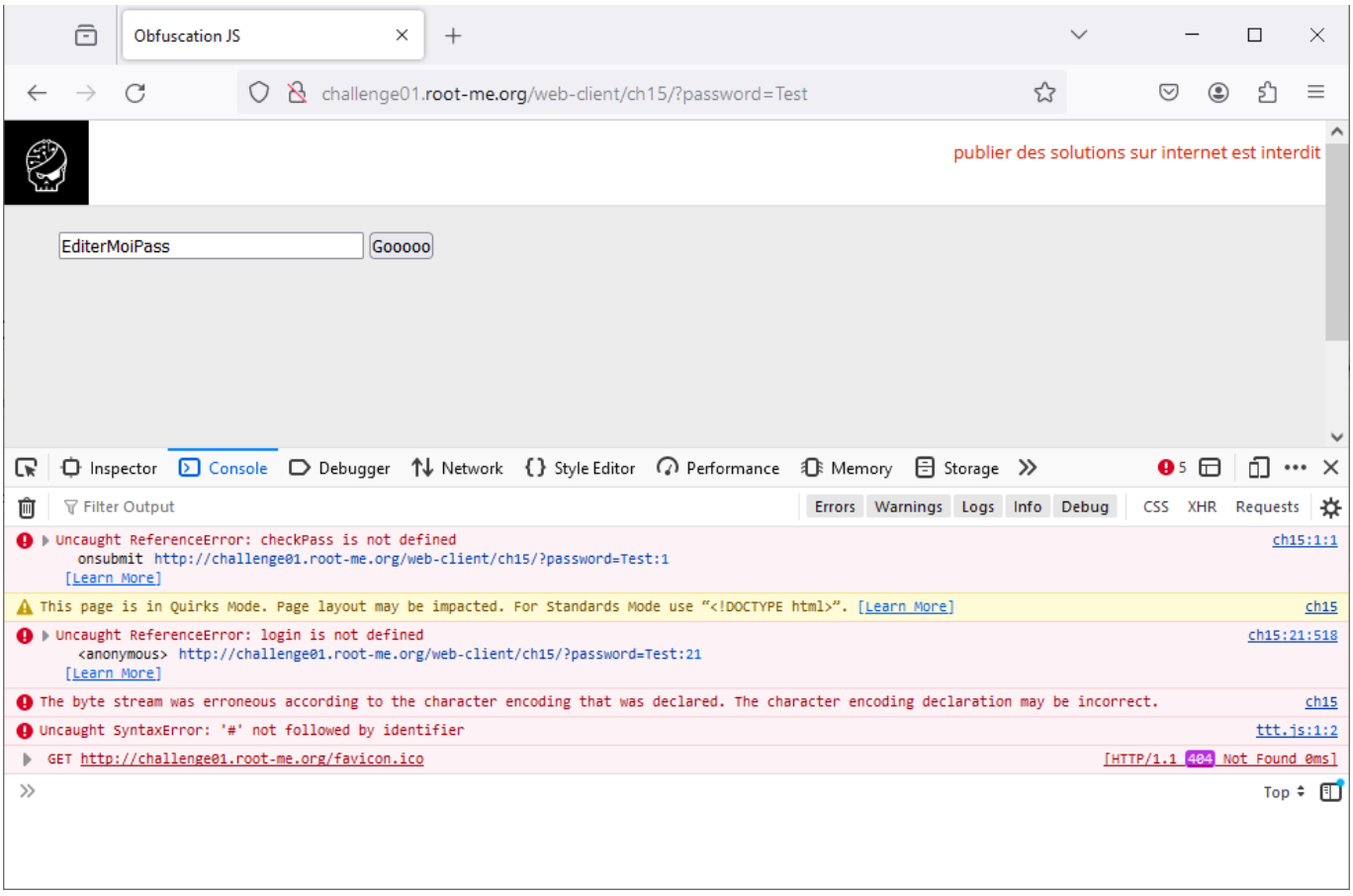
By accessing the challenge page with firefox, we can see a single textbox and a button:



After changing the text and clicking the button, the page redirect to itself with the query `?password=[Text entered]`



Using the build-in developper tools of the browser, we can see that there are multiple errors in the page, and several functions are not defined:



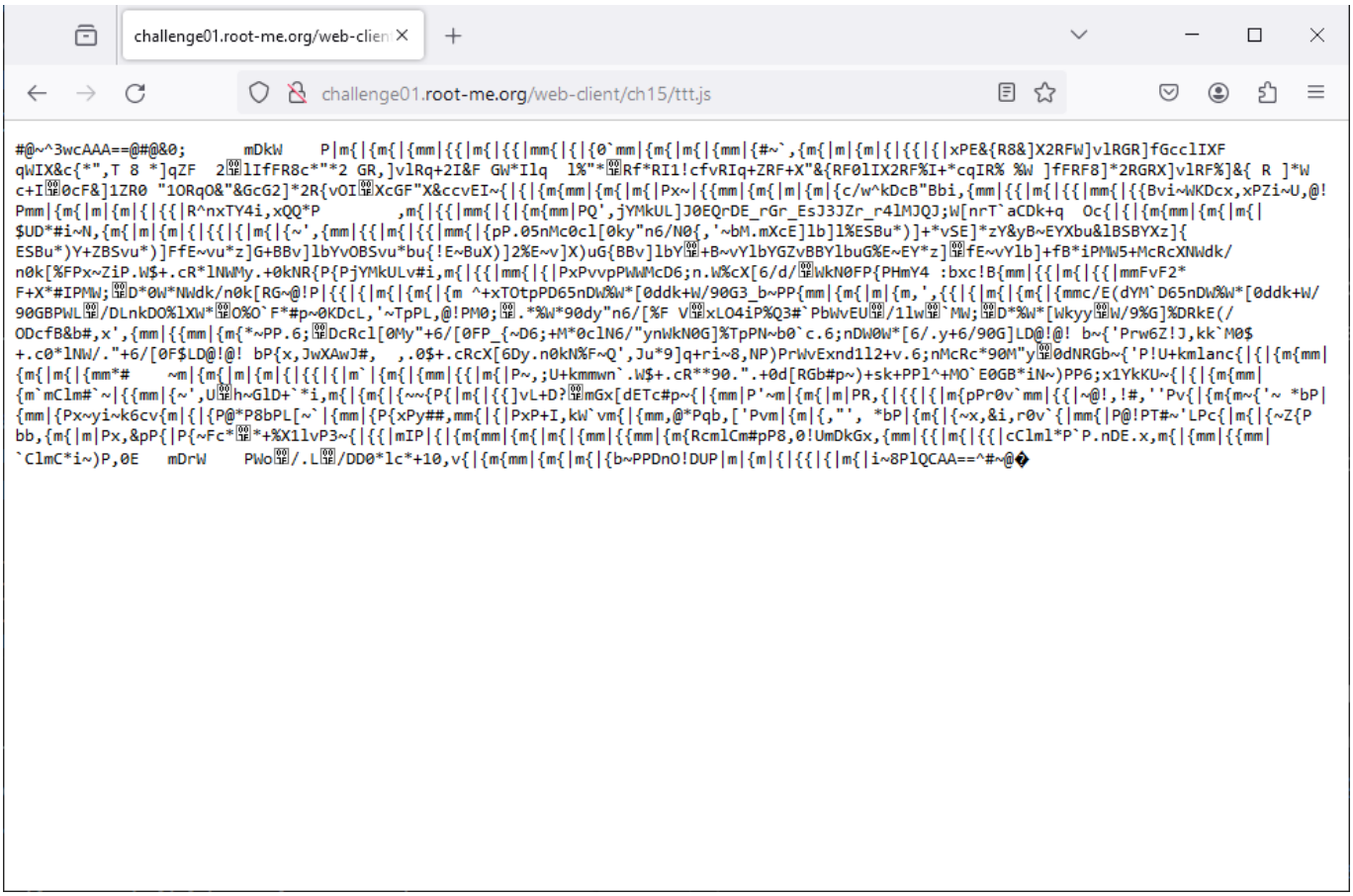
We have to look into the html source code which looks like this:

```
<html>
<head>
    <title>Obfuscation JS</title>
<!--
Obfuscation
.Niveau : Très Difficile
.By Hel0ck
.The mission :
    Retrouvez le password :)
    Vous allez en baver HAHAHAHAHHAHHHAHAHHAHAHAHAHHA :x

    /!\ ------------------------------------------------
    INFO !
    Pour que le script marche, vous devez d'abord trouver comment décoder le
ttt.js
    Ce n'est pas trop difficile, faut chercher :D
    ------------------------------------------------------ /!\

    You need my help ? IRC : irc.root-me.org #root-me
-->
<script type="text/javascript">
    _ = 'function recfn0()......
</script> <script type="text/javascript" src="ttt.js"></script>
</head>
<body><link rel='stylesheet' property='stylesheet' id='s' type='text/css'
href='/template/s.css' media='all' /><iframe id='iframe' src='https://www.root-
me.org/?page=externe_header'></iframe>
    <form name="login" id="login" action="" method="get" onsubmit="checkPass()">
        <input id="password" name="password"  value="EditerMoiPass" size="30"
maxlength="30" />
        <input type="submit" value="Gooooo">
    </form>
</body>
</html>
```

We can see that the html is pretty normal, but the script isn't: it is divided into two parts, the first part is very long and obfuscated(omitted here), the second part is in the file `ttt.js`.
We also get the hint that we have to first decode `ttt.js`

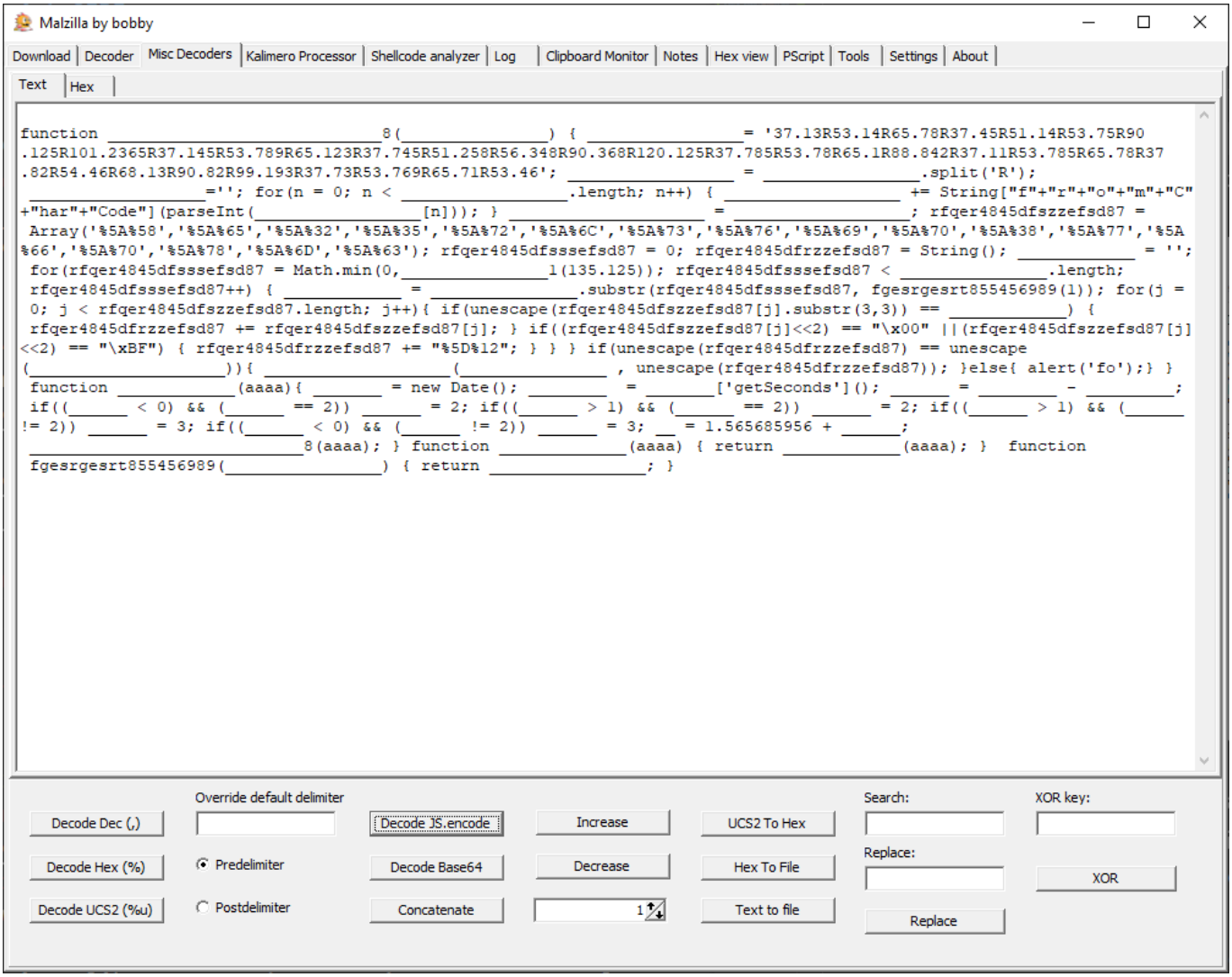
Decode ttt.js

Using the URL `http://challenge01.root-me.org/web-client/ch15/ttt.js`, we found out that it is indeed encoded into a character string:



After a lot of research and trial and error, we stumbled upon a encoding method called JScript.Encode :
https://en.wikipedia.org/wiki/JScript.Encode
which is reverse engineered : http://virtualconspiracy.com/content/articles/breaking-screnc

We used Malzilla tool (available here https://malzilla.org) to decode the script.



Now the 2 parts of javascript can be put together. But still, the script is broken : we got errors when we try to execute it

Clean up

The script is messed up and we need to beautify the code using https://beautifier.io
The result:

```
_ = 'function recfn0(){ return 0;}';
___ = '';
__ = 0;
eval(_);
for (__________ = 0; __________ < parseInt(100.15786224552145); __________++) {
    n = __________ + 1;
    ___ += 'function recfn' + n + '(){ return recfn' + __________ + '();}';
}
eval(___);
eval(recfn100());
____ = this;
for (_____ in ____) {
    if (_____.length == parseInt(10.125625)) {
```

```
        if (_____.charCodeAt(0) == Math.min(115, 12654)) {
            if (_____.charCodeAt(9) == 116) break;
        }
    }
}
var ___________ = new Date();
_________ = ___________['getSeconds']();
____[_____]('_____________("' + login.password.value + '")',
_____________(2000));

function _____________(fgseg87857878562) {
    var fgsog87857878562, fgspg87857878562, fgspg87857878561, fgspg87817878562, d,
r, fgspg87817878162;
    fgsog87857878562 = fgseg87857878562;
    fgspg87857878562 += fgsog87857878562 >> (fgspg87817878562 - fgsog87857878562);
    fgspg87857878561 -= fgspg87817878562 + fgsog87857878562 + 1337;
    fgspg87817878162 = r ^ fgspg87857878561 + fgsog87857878562;
    return fgsog87857878562;
}

function fgspg87857878561(fgspd87857878561) {
    return fgspd87857878561;
}

function _____________1(PKDFjidfjezif8756265) {
    _________________ = PKDFjidfjezif8756265;
    _________________ = 145;
    if (_________________ == 1) {
        return parseInt(1.256);
    }
    if (_________________ == 2) {
        return parseInt(2.145);
    }
    if (_________________ == 3) {
        return parseInt(3.145);
    }
    if (_________________ > 3) {
        _________________ = _________________ + _________________ -
_________________;
        return _________________;
    }
    return _________________;
}

function ___________________(_____________, _____________) {
    ___________ = '8aZ{E$+rT yU}1#2(IOP<qs,DFg.)H*Jk~L6M7]W;X%VxB:N!^-03/9[4&5|"?
Kz';
    _________ = escape(_____________ + _____________ + "eDer");
    output = "";
    var qrfqfqfq844568, qrfqfqfq814568, qsfqfqfq814568 = "";
    var _3, _1, _2, _0 = "";
    ______ = 0;
    _____________ = ___________;
    do {
```

```
        qrfqfqfq844568 = _________.charCodeAt(______++);
        qrfqfqfq814568 = _________.charCodeAt(______++);
        qsfqfqfq814568 = _________.charCodeAt(______++);
        _3 = qrfqfqfq844568 >> _______________1(2);
        _1 = ((qrfqfqfq844568 & _______________1(3)) << 4) | (qrfqfqfq814568 >>
_______________1(4));
        _2 = ((qrfqfqfq814568 & _______________1(15)) << _______________1(2)) |
(qsfqfqfq814568 >> 6);
        _0 = qsfqfqfq814568 & 63;
        if (isNaN(qrfqfqfq814568)) {
            _2 = _0 = 64;
        } else if (isNaN(qsfqfqfq814568)) {
            _0 = 64;
        }
        output = output + _____________.charAt(_3) + _____________.charAt(_1) +
_____________.charAt(_2) + _____________.charAt(_0);
        if (arguments.callee.toString().length != 1731) {
            output = "ESF0 ('7p(:5J";
        }
        qrfqfqfq844568 = qrfqfqfq814568 = qsfqfqfq814568 = "";
        _3 = _1 = _2 = _0 = "";
    } while (______ < _________.length);
    if (fgspg87857878561235(output, __) ==
fgspg87857878561235("}8□iH□5:}Ypi}*VL}", 13) + fgspg87857878561("^2d2S*,~") + ":"
+ String["fr" + "om" + "C" + "harCode"](74, 76, 69, 83, 70, 48, 32, 40, 39, 55,
112, 40, 44, 53, 74, 39, 60, 44, 50, 112, 114, 70, 69, 47, 87)) {
        window.location.href = _______________ + ".php";
    } else {
        alert("MOUHAHAHAHHAHAHAHAHA");
    }
}

function fgspg87857878561235(q45qsdfus8465dfsfgth, __) {
    q45qsdfzf8465dfsfgth = "";
    q45qsdfzf8465dfsfguh = "";
    _______________ = 0000000000;
    for (_______________; _______________ < 13 - 2; _______________++) {
        q45qsdfzf8465dfsfgth = q45qsdfus8465dfsfgth.length ^ __;
        if (q45qsdfzf8465dfsfgth != "") q45qsdfzf8465dfsfgth = "";
    }
    q45qsdfzs8465dfsfgth = q45qsdfzf8465dfsfgth;
    q45qsdfzf8465dfsfguh = q45qsdfzs8465dfsfgth;
    for (i = 0; i < q45qsdfus8465dfsfgth.length; i++) {
        if (q45qsdfzs8465dfsfgth == "ESF0 ('7p(,5J')") {
            q45qsdfzs8465dfsfgth +=
String[unescape("66%72%6f%6d%43%68%61%72%43%6f%64%65")](__ ^
q45qsdfus8465dfsfgth.charCodeAt(i) + 12);
        } else {
            q45qsdfzs8465dfsfgth += String.fromCharCode(__ ^
q45qsdfus8465dfsfgth.charCodeAt(i));
        }
    }
    if (arguments.callee.toString().length != 1169) {
        q45qsdfzs8465dfsfgth = "ESF0 ('7p(:5J";
```

```
    }
    if (q45qsdfzf8465dfsfguh == "sfeze5825qsde8rfq--") {
        for (______ = 0; ______ > -1; ______++) {
            q45qsdfzf8465dfsfguh = "\x6e\x6f\x6f\x62\x20\x6e\x6f\x6f\x62";
        }
    }
    return q45qsdfzs8465dfsfgth;
}

function _____________________________8(_________________) {
    ________________ =
'37.13R53.14R65.78R37.45R51.14R53.75R90.125R101.2365R37.145R53.789R65.123R37.745R5
1.258R56.348R90.368R120.125R37.785R53.78R65.1R88.842R37.11R53.785R65.78R37.82R54.4
6R68.13R90.82R99.193R37.73R53.769R65.71R53.46';
    ________________ = ________________.split('R');
    ________________ = '';
    for (n = 0; n < ________________.length; n++) {
        ____________________ += String["f" + "r" + "o" + "m" + "C" + "har" + "Code"]
(parseInt(________________[n]));
    }
    ____________________ = ____________________;
    rfqer4845dfszzefsd87 = Array('%5A%58', '%5A%65', '%5A%32', '%5A%35', '%5A%72',
'%5A%6C', '%5A%73', '%5A%76', '%5A%69', '%5A%70', '%5A%38', '%5A%77', '%5A%66',
'%5A%70', '%5A%78', '%5A%6D', '%5A%63');
    rfqer4845dfsssefsd87 = 0;
    rfqer4845dfrzzefsd87 = String();
    ____________ = '';
    for (rfqer4845dfsssefsd87 = Math.min(0, _____________1(135.125));
rfqer4845dfsssefsd87 < _____________.length; rfqer4845dfsssefsd87++) {
        ____________ = _____________.substr(rfqer4845dfsssefsd87,
fgesrgesrt855456989(1));
        for (j = 0; j < rfqer4845dfszzefsd87.length; j++) {
            if (unescape(rfqer4845dfszzefsd87[j].substr(3, 3)) == ____________) {
                rfqer4845dfrzzefsd87 += rfqer4845dfszzefsd87[j];
            }
            if ((rfqer4845dfszzefsd87[j] << 2) == "\x00" ||
(rfqer4845dfszzefsd87[j] << 2) == "\xBF") {
                rfqer4845dfrzzefsd87 += "%5D%12";
            }
        }
    }
    if (unescape(rfqer4845dfrzzefsd87) == unescape(____________________)) {
        ____________________(_____________, unescape(rfqer4845dfrzzefsd87));
    } else {
        alert('fo');
    }
}

function ____________(aaaa) {
    ________ = new Date();
    ________ = ________['getSeconds']();
    ______ = ________ - _________;
    if ((______ < 0) && (______ == 2)) ______ = 2;
    if ((______ > 1) && (______ == 2)) ______ = 2;
```

```
    if ((______ > 1) && (______ != 2)) ______ = 3;
    if ((______ < 0) && (______ != 2)) ______ = 3;
    __ = 1.565685956 + ______;
    _____________________________8(aaaa);
}

function _____________(aaaa) {
    return _____________(aaaa);
}

function fgesrgesrt855456989(________________) {
    return ________________;
}
```

We tried online deobfuscaters but they didn't do a lot. So we have to do it by hand: Rename function and variable names, simplify constant functions like `parseInt(constant)`, remove several useless functions, such as those who only returns the input, or those who are never called. There are also a lot of if conditions that won't trigger, and variables that are not used. After all this, the code is readable:

```
globalvar = 0;

evalstring = '';
function recfn0(){ return 0;}
for (i = 0; i < 100; i++) {
    n = i + 1;
    evalstring += 'function recfn' + n + '(){ return recfn' + i + '();}';
}
eval(evalstring);
eval(recfn100());

for (obj in this) {
    if (obj.length == 10) {
        if (obj.charCodeAt(0) == 115) {
            if (obj.charCodeAt(9) == 116) break;
        }
    }
}
var date1 = new Date();
seconds1 = date1['getSeconds']();
this[obj]('func4("' + login.password.value + '")', 2000);

function func1(input1, input2) {
    str1 = '8aZ{E$+rT yU}1#2(IOP<qs,DFg.)H*Jk~L6M7]W;X%VxB:N!^-03/9[4&5|"?Kz';
    str2 = escape(input2 + input1 + "eDer");
    output = "";
    var a, b, c = "";
    var _3, _1, _2, _0 = "";
    cnt = 0;
    do {
        a = str2.charCodeAt(cnt++);
        b = str2.charCodeAt(cnt++);
```

```
        c = str2.charCodeAt(cnt++);
        _3 = a >> 2;
        _1 = ((a & 3) << 4) | (b >> 4);
        _2 = ((b & 15) << 2) | (c >> 6);
        _0 = c & 63;
        if (isNaN(b)) {
            _2 = _0 = 64;
        } else if (isNaN(c)) {
            _0 = 64;
        }
        output = output + str1.charAt(_3) + str1.charAt(_1) + str1.charAt(_2) +
str1.charAt(_0);
        if (arguments.callee.toString().length != 1731) {
            output = "ESF0 ('7p(:5J";
        }
        a = b = c = "";
        _3 = _1 = _2 = _0 = "";
    } while (cnt < str2.length);
    if (func2(output, globalvar) == func2("}8□iH□5:}Ypi}*VL}", 13) + "^2d2S*,~" +
":" + String["fr" + "om" + "C" + "harCode"](74, 76, 69, 83, 70, 48, 32, 40, 39,
55, 112, 40, 44, 53, 74, 39, 60, 44, 50, 112, 114, 70, 69, 47, 87)) {
        window.location.href = input1 + ".php";
    } else {
        alert("MOUHAHAHAHHAHAHAHAHA");
    }
}

function func2(input1, input2) {
    a = "";
    b = "";
    for (i = 0; i < 11; i++) {
        a = input1.length ^ input2;
        if (a != "") a = "";
    }
    c = a;
    b = c;
    for (i = 0; i < input1.length; i++) {
        if (c == "ESF0 ('7p(,5J')") {
            c += String[unescape("66%72%6f%6d%43%68%61%72%43%6f%64%65")](input2 ^
input1.charCodeAt(i) + 12);
        } else {
            c += String.fromCharCode(input2 ^ input1.charCodeAt(i));
        }
    }
    if (arguments.callee.toString().length != 1169) {
        c = "ESF0 ('7p(:5J";
    }
    if (b == "sfeze5825qsde8rfq--") {
        for (i = 0; i > -1; i++) {
            b = "\x6e\x6f\x6f\x62\x20\x6e\x6f\x6f\x62";
        }
    }
    return c;
}
```

```
function func3(input) {
    str1 =
'37.13R53.14R65.78R37.45R51.14R53.75R90.125R101.2365R37.145R53.789R65.123R37.745R5
1.258R56.348R90.368R120.125R37.785R53.78R65.1R88.842R37.11R53.785R65.78R37.82R54.4
6R68.13R90.82R99.193R37.73R53.769R65.71R53.46';
    splited = str1.split('R');
    str2 = '';
    for (n = 0; n < splited.length; n++) {
        str2 += String["f" + "r" + "o" + "m" + "C" + "har" + "Code"]
(parseInt(splited[n]));
    }
    a = Array('%5A%58', '%5A%65', '%5A%32', '%5A%35', '%5A%72', '%5A%6C',
'%5A%73', '%5A%76', '%5A%69', '%5A%70', '%5A%38', '%5A%77', '%5A%66', '%5A%70',
'%5A%78', '%5A%6D', '%5A%63');
    str3 = String();
    str4 = '';
    for (i = 0; i < input.length; i++) {
        str4 = input.substr(i, 1);
        for (j = 0; j < a.length; j++) {
            if (unescape(a[j].substr(3, 3)) == str4) {
                str3 += a[j];
            }
            if ((a[j] << 2) == "\x00" || (a[j] << 2) == "\xBF") {
                str3 += "%5D%12";
            }
        }
    }
    if (unescape(str3) == unescape(str2)) {
        func1(input, unescape(str3));
    } else {
        alert('fo');
    }
}

function func4(aaaa) {
    date2 = new Date();
    seconds2 = date2['getSeconds']();
    delta = seconds2 - seconds1;
    if (delta > 2) delta = 3;
    if (delta < 0) delta = 3;
    globalvar = 1.565685956 + delta;
    func3(aaaa);
}
```

Dissect the code

The easy part is done, now we have to read and understand the code to progress.

Some return functions

```
evalstring = '';
function recfn0(){ return 0;}
for (i = 0; i < 100; i++) {
    n = i + 1;
    evalstring += 'function recfn' + n + '(){ return recfn' + i + '();}';
}
eval(evalstring);
eval(recfn100());
```

This part of code creates 100 functions that returns 0. However, these functions are never called so we can safely delete those.

A object in this

```
for (obj in this) {
    if (obj.length == 10) {
        if (obj.charCodeAt(0) == 115) {
            if (obj.charCodeAt(9) == 116) break;
        }
    }
}
var date1 = new Date();
seconds1 = date1['getSeconds']();
this[obj]('func4("' + login.password.value + '")', 2000);
```

This part finds a object of `this` (window) and called it. By running this part of the script on firefox, we found out it is the `setTimeout` object. It means that `func4(login.password.value)` will be executed 2 seconds after the page is loaded. Even though `login` is not well defined in the html, we guess that `login.password.value` should be the text we put in the textbox.

func4

```
function func4(aaaa) {
    date2 = new Date();
    seconds2 = date2['getSeconds']();
    delta = seconds2 - seconds1;
    if (delta > 2) delta = 3;
    if (delta < 0) delta = 3;
    globalvar = 1.565685956 + delta;
    func3(aaaa);
}
```

All `func4` does is to calculate `globalvar` and call `func3`. Delta is always 2 (2 seconds of waiting time) and the globalvar is constant.

func3

```
str1 =
'37.13R53.14R65.78R37.45R51.14R53.75R90.125R101.2365R37.145R53.789R65.123R37.745R5
1.258R56.348R90.368R120.125R37.785R53.78R65.1R88.842R37.11R53.785R65.78R37.82R54.4
6R68.13R90.82R99.193R37.73R53.769R65.71R53.46';
splited = str1.split('R');
str2 = '';
for (n = 0; n < splited.length; n++) {
    str2 += String["f" + "r" + "o" + "m" + "C" + "har" + "Code"]
(parseInt(splited[n]));
}
a = Array('%5A%58', '%5A%65', '%5A%32', '%5A%35', '%5A%72', '%5A%6C', '%5A%73',
'%5A%76', '%5A%69', '%5A%70', '%5A%38', '%5A%77', '%5A%66', '%5A%70', '%5A%78',
'%5A%6D', '%5A%63');
str3 = String();
str4 = '';
```

This first part of `func3` is constant and can be calculated. `array a = {ZX,Ze,Z2,Z5,Zr,Zl,Zs,Zv,Zi,Zp,Z8,Zw,Zf,Zp,Zx,Zm,Zc}` if unescaped. The reset of `func3`:

```
for (i = 0; i < input.length; i++) {
    str4 = input.substr(i, 1);
    for (j = 0; j < a.length; j++) {
        if (unescape(a[j].substr(3, 3)) == str4) {
            str3 += a[j];
        }
        if ((a[j] << 2) == "\x00" || (a[j] << 2) == "\xBF") {
            str3 += "%5D%12";
        }
    }
}
if (unescape(str3) == unescape(str2)) {
    func1(input, unescape(str3));
} else {
    alert('fo');
}
```

After further inspection, we found out `func3` is a check for the password: we iterate through all the character of the password, if it is one of the second character of array a, we put `'Z'+(the character)` to a output string. Then it checks if the output string equals to `"Z5ZeZ8ZxZXZmZcZ5"`, and if not, the script ends with a single alert.
We can deduce that for the password to pass this test, it has to have a non-continuous sub string `"5e8xXmc5"`, and the rest of the password contains no following characters `258Xcefilmprsvwx` (in ASCII order)

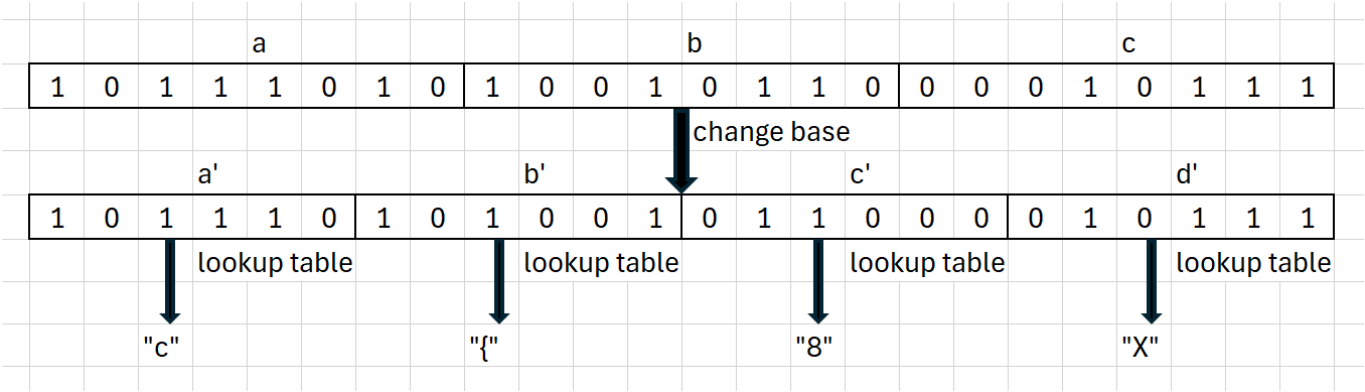
func1

```

```

```
function func1(input1, input2) {
    str1 = '8aZ{E$+rT yU}1#2(IOP<qs,DFg.)H*Jk~L6M7]W;X%VxB:N!^-03/9[4&5|"?Kz';
    str2 = escape(input2 + input1 + "eDer");
    output = "";
    var a, b, c = "";
    var _3, _1, _2, _0 = "";
    cnt = 0;
    do {
        a = str2.charCodeAt(cnt++);
        b = str2.charCodeAt(cnt++);
        c = str2.charCodeAt(cnt++);
        _3 = a >> 2;
        _1 = ((a & 3) << 4) | (b >> 4);
        _2 = ((b & 15) << 2) | (c >> 6);
        _0 = c & 63;
        if (isNaN(b)) {
            _2 = _0 = 64;
        } else if (isNaN(c)) {
            _0 = 64;
        }
        output = output + str1.charAt(_3) + str1.charAt(_1) + str1.charAt(_2) +
str1.charAt(_0);
        if (arguments.callee.toString().length != 1731) {
            output = "ESF0 ('7p(:5J";
        }
        a = b = c = "";
        _3 = _1 = _2 = _0 = "";
    } while (cnt < str2.length);
    if (func2(output, globalvar) == func2("}8□iH□5:}Ypi}*VL}", 13) + "^2d2S*,~" +
":" + String["fr" + "om" + "C" + "harCode"](74, 76, 69, 83, 70, 48, 32, 40, 39,
55, 112, 40, 44, 53, 74, 39, 60, 44, 50, 112, 114, 70, 69, 47, 87)) {
        window.location.href = input1 + ".php";
    } else {
        alert("MOUHAHAHAHHAHAHAHAHA");
    }
}
```

At the end of `func3`, `func1` is called with the first argument the entered password and the second argument the constant string `"Z5ZeZ8ZxZXZmZcZ5"`. By simlating the bitwise operations , we found out the first part of the function is basically a encoder that converts char string into a custom base64 code. Instead of using normal base64 sequence, it uses `8aZ{E$+rT yU}1#2(IOP<qs,DFg.)H*Jk~L6M7]W;X%VxB:N!^-03/9[4&5|"?Kz` as its character table. A simple illustration

(not real values):



However, `if (arguments.callee.toString().length != 1731)`, which means if the whole function declaration isn't the correct size, the output resets to a specific string each time. The original size of the obfuscated function is 1477, which is different from 1731. Some posts on Internet mentioned that Firefox might optimize the code before calling `function.toString()`, so we tried to use the deprecated Internet Explorer. Now that Microsoft is promoting Edge browser, we have to open the Internet Explorer using a VBS script:

```
Set objIE = CreateObject("InternetExplorer.Application")
objIE.Navigate "http://challenge01.root-me.org/web-client/ch15/"
objIE.Visible = 1
```

We verified the size of the function, and unfortunately it is the same result as given by Firefox. We are safe to say that the output string will be overridden if the original script is executed. We are curious to find out if it is intend or not.

func2

```
function func2(input1, input2) {
    a = "";
    b = "";
    for (i = 0; i < 11; i++) {
        a = input1.length ^ input2;
        if (a != "") a = "";
    }
    c = a;
    b = c;
    for (i = 0; i < input1.length; i++) {
        if (c == "ESF0 ('7p(,5J')") {
            c += String[unescape("66%72%6f%6d%43%68%61%72%43%6f%64%65")](input2 ^
input1.charCodeAt(i) + 12);
        } else {
            c += String.fromCharCode(input2 ^ input1.charCodeAt(i));
        }
    }
    if (arguments.callee.toString().length != 1169) {
        c = "ESF0 ('7p(:5J";
    }
```

```
    if (b == "sfeze5825qsde8rfq--") {
        for (i = 0; i > -1; i++) {
            b = "\x6e\x6f\x6f\x62\x20\x6e\x6f\x6f\x62";
        }
    }
    return c;
}
```

At the end of `func1`, `func2` is called. The second param is either 3 or 13, and is only used in bitwize xor. The escaped string is evidently `"fromCharCode"`, but it missed a `%`. After deleting useless parts of the code, it is clear that `func2` take all the characters of the first argument and xor it with the second argument. The only exception is `if (c == "ESF0 ('7p(,5J')")` we have to change a little bit.
However, everything seemed to be quite useless, as it checks the length of the function as well, and all it does now is to return a constant sting.

The goal

```
if (func2(output, 3) == func2("}8□iH□5:}Ypi}*VL}", 13) + "^2d2S*,~:JLESF0
('7p(,5J'<,2prFE/W") {
    window.location.href = input + ".php";
}
```

Finally, if all the tests are passed, the script will redirect the page to `(password).php`. We guess that this file will give us the answer of the problem. Now the whole program make more sense: the steps to check length of function might be broken during the obfuscation process, and the whole javascript code isn't meant to be executed on the web page; Instead, we just need to find out a password that checks all the tests, and go directly to the php page. We can thus ignore those function length checks.
This is the code that is left after all simplifications:

```
func3("testpassword");

function func1(input) {
    str1 = '8aZ{E$+rT yU}1#2(IOP<qs,DFg.)H*Jk~L6M7]W;X%VxB:N!^-03/9[4&5|"?Kz';
    str2 = escape("Z5ZeZ8ZxZXZmZcZ5" + input + "eDer");
    output = "";
    cnt = 0;

    do {
        var a, b, c = "";
        var _3, _1, _2, _0 = "";
        a = str2.charCodeAt(cnt++);
        b = str2.charCodeAt(cnt++);
        c = str2.charCodeAt(cnt++);
        _3 = a >> 2;
        _1 = ((a & 3) << 4) | (b >> 4);
        _2 = ((b & 15) << 2) | (c >> 6);
        _0 = c & 63;
        if (isNaN(b)) {
```

```
            _2 = _0 = 64;
        } else if (isNaN(c)) {
            _0 = 64;
        }
        output = output + str1.charAt(_3) + str1.charAt(_1) + str1.charAt(_2) +
str1.charAt(_0);
    } while (cnt < str2.length);

    if (func2(output, 3) == func2("}8□iH□5:}Ypi}*VL}", 13) + "^2d2S*,~:JLESF0
('7p(,5J'<,2prFE/W") {
        window.location.href = input + ".php";
    } else {
        console.log("MOUHAHAHAHHAHAHAHAHA");
    }
}

function func2(input1, input2) {
    c = "";
    for (i = 0; i < input1.length; i++) {
        if (c == "ESF0 ('7p(,5J')") {
            c += String.fromCharCode(input2 ^ input1.charCodeAt(i) + 12);
        } else {
            c += String.fromCharCode(input2 ^ input1.charCodeAt(i));
        }
    }
    return c;
}

function func3(input) {
    a = Array('%5A%58', '%5A%65', '%5A%32', '%5A%35', '%5A%72', '%5A%6C',
'%5A%73', '%5A%76', '%5A%69', '%5A%70', '%5A%38', '%5A%77', '%5A%66', '%5A%70',
'%5A%78', '%5A%6D', '%5A%63');
    ans = '';
    c = '';
    for (i = 0; i < input.length; i++) {
        c = input.substr(i, 1);
        for (j = 0; j < a.length; j++) {
            if (unescape(a[j].substr(3, 3)) == c) {
                ans += a[j];
            }
        }
    }
    if (unescape(ans) == "Z5ZeZ8ZxZXZmZcZ5") {
        func1(input);
    } else {
        console.log('fo');
    }
}
```

Calculate the password

We start working backwards:
We have to have `func2(string, 3) == "p5rdEr87pT}dp'[Ap^2d2S*,~:JLESF0 ('7p(,5J'<,2prFE/W"`
And since xor is reversable by simply doing another xor, we calculate the string with `func2("p5rdEr87pT}dp'[Ap^2d2S*,~:JLESF0 ('7p(,5J'<,2prFE/W", 3)` which gave us `s6qgFq;4sW~gs$XBs]1g1P)/}9IOFPE3#+$4s+/6I$?/1sqEF,T`. Fortunately, we never have to deal with the special case while doing the xor.

Then we need to decode it back to char string. Writing a decoder similar to the encoder does the trick:

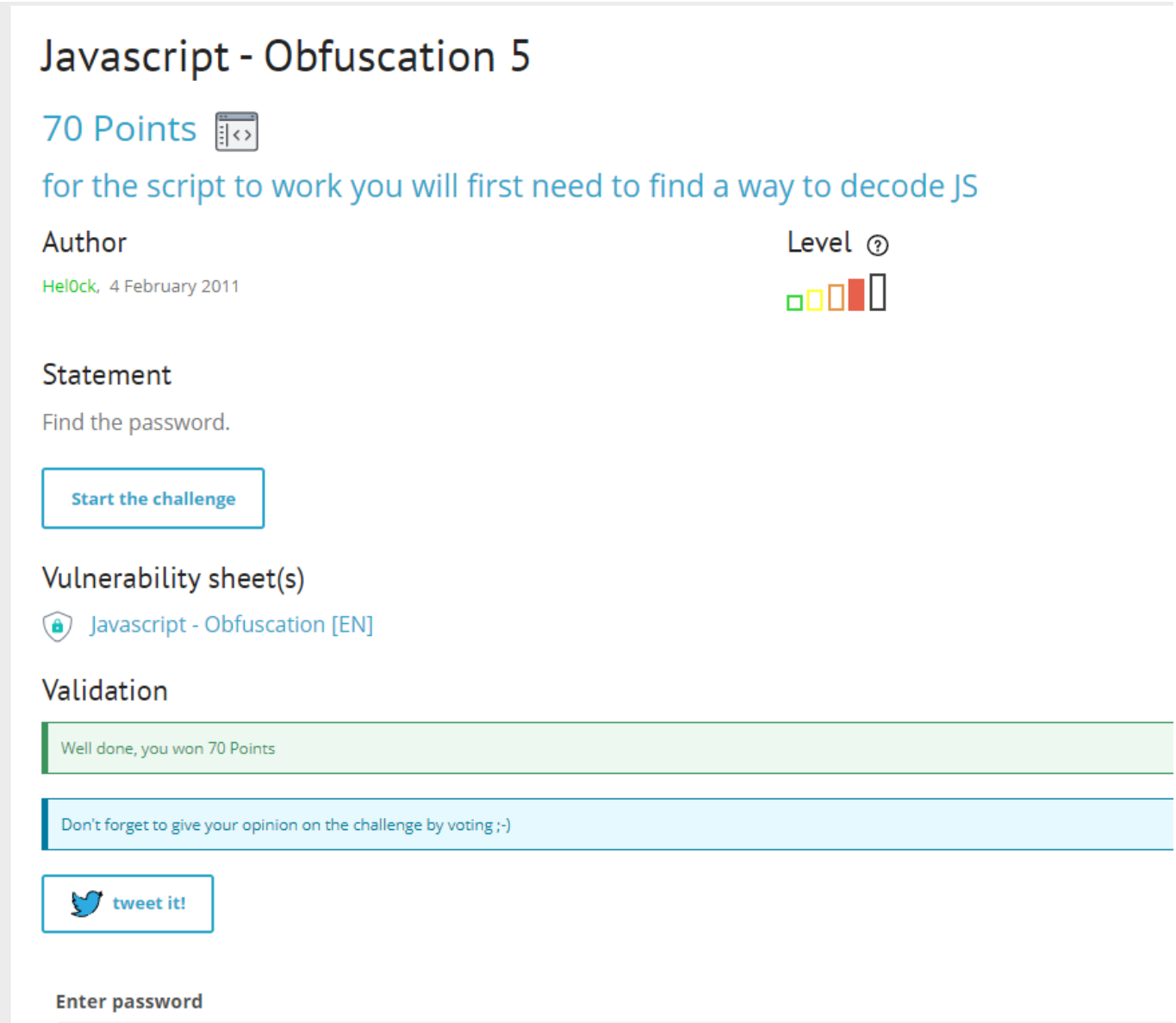
```
function func1_decode() {
    str1 = '8aZ{E$+rT yU}1#2(IOP<qs,DFg.)H*Jk~L6M7]W;X%VxB:N!^-03/9[4&5|"?Kz';
    func1_output = "s6qgFq;4sW~gs$XBs]1g1P)/}9IOFPE3#+$4s+/6I$?/1sqEF,T";
    text = "";
    cnt = 0;

    do {
        var a, b, c = "";
        var code1, code2, code3, code4 = "";

        code1 = str1.indexOf(func1_output.charAt(cnt++));
        code2 = str1.indexOf(func1_output.charAt(cnt++));
        code3 = str1.indexOf(func1_output.charAt(cnt++));
        code4 = str1.indexOf(func1_output.charAt(cnt++));

        a = (code1 << 2) | (code2 >> 4);
        b = ((code2 & 15) << 4) | (code3 >> 2);
        c = ((code3 & 3) << 6) | code4;
        text += String.fromCharCode(a) + String.fromCharCode(b) +
String.fromCharCode(c);
    } while (cnt < func1_output.length);

    console.log(text);
}
```

It gave us the result `Z5ZeZ8ZxZXZmZcZ5753dRe148axXmcD_u5eDer`.
The last character is just from the edge case and is ignored. We immediately recognized the pattern, and by truncating the head and tail, this is what is left: `753dRe148axXmcD_u5`
Luckily, this string checks the first test, meaning that it should be the correct password!

Capture the flag

Visiting http://challenge01.root-me.org/web-client/ch15/753dRe148axXmcD_u5.php gave us a blank page, but it also mean that the file is there (or there will be a 404 error). Hence, `753dRe148axXmcD_u5` must be the

password and we captured the flag!



Counter measures

The most evident and effective counter measure for this "attack" is NEVER put sensitive code in the public. Verification of the password should be done on the server side, and we should use a encryption method to transfer data.
If we think about the question: Why would anyone obfuscate their code that is given to others? It may deter some attacks, but if the attacker is determined, it is always reversable. All it does is to make reverse engineering harder.
We also found out that obfuscation is commonly used in malware, to evade detection from antivirus programs, and hide vulnerabilities that it exploits.
In conclusion, do not put youself in a situation where you have to obfuscate your code, but if you have to, just make it harder.