

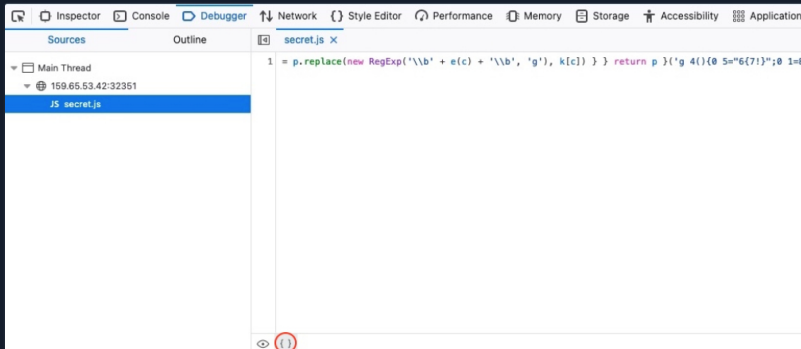
Deobfuscation

Now that we understand how code obfuscation works let's start our learning towards deobfuscation. Just as there are tools to obfuscate code automatically, there are tools to beautify and deobfuscate the code automatically.

Beautify

We see that the current code we have is all written in a single line. This is known as **Minified JavaScript** code. In order to properly format the code, we need to **Beautify** our code. The most basic method for doing so is through our **Browser Dev Tools**.

For example, if we were using Firefox, we can open the browser debugger with **CTRL+SHIFT+Z**, and then click on our script **secret.js**. This will show the script in its original formatting, but we can click on the **{ }** button at the bottom, which will **Pretty Print** the script into its proper JavaScript formatting:

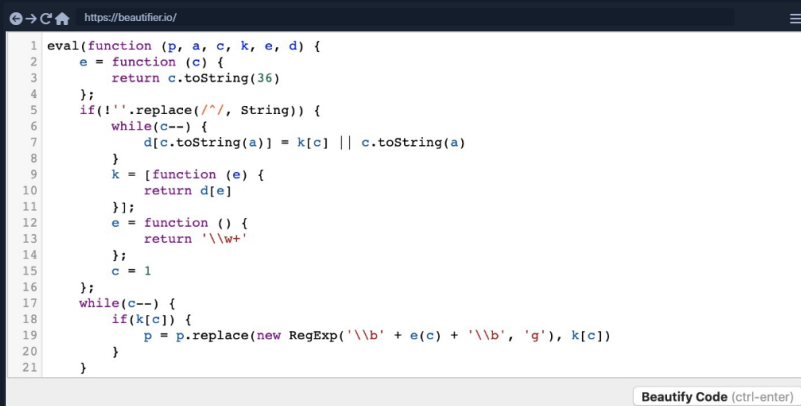
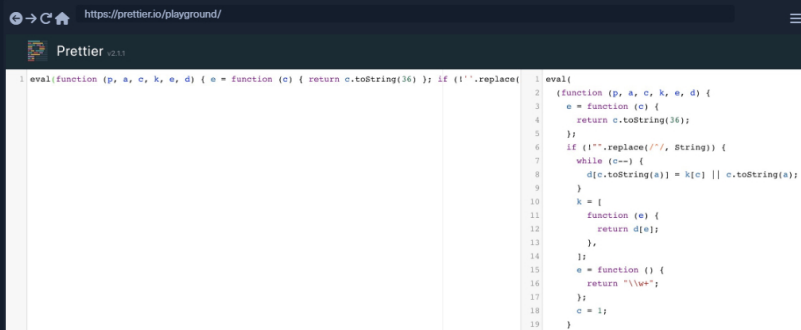


Furthermore, we can utilize many online tools or code editor plugins, like **Prettier** or **Beautifier**. Let us copy the **secret.js** script:

Code: **javascript**

```
eval(function (p, a, c, k, e, d) { e = function (c) { return c.toString(36) }; if (!''.replace(/^/, String)) { while (c
```

We can see that both websites do a good job in formatting the code:



However, the code is still not very easy to read. This is because the code we are dealing with was not only minified but obfuscated as well. So, simply formatting or beautifying the code will not be enough. For that, we will require tools to deobfuscate the code.

Deobfuscate

We can find many good online tools to deobfuscate JavaScript code and turn it into something we can understand. One good tool is **UnPacker**. Let's try copying our above-obfuscated code and run it in UnPacker by clicking the **UnPack** button.

[Cheat Sheet](#)[Go to Questions](#)

Table of Contents

Introduction

[Introduction](#)[Source Code](#)

Obfuscation

[Code Obfuscation](#)[Basic Obfuscation](#)[Advanced Obfuscation](#)[Deobfuscation](#)

Deobfuscation Examples

[Code Analysis](#)[HTTP Requests](#)[Decoding](#)

Skills Assessment

[Skills Assessment](#)[Summary](#)

My Workstation

OFFLINE

[Start Instance](#)

00 / 1 spawns left

Tip: Ensure you do not leave any empty lines before the script, as it may affect the deobfuscation process and give inaccurate results.

<https://matthewfl.com/unPacker.html>

```
eval(function (p, a, c, k, e, d) { e = function (c) { return c.toString(36) }; if (!''.replace(/^/, String)) { while (c--) { d[c.toString(a)] = k[c] || c.toString(a) } k = [function (e) { return d[e] }]; e = function () { return '\\w+' }; c = 1 }; while (c--) { if (k[c]) { p = p.replace(new RegExp('\\b' + e(c) + '\\b', 'g'), k[c]) } } return p }('g 4(){0 5="6{7!}";0 1=8 a();0 2="/9.c";1.d("e",2,f);1.b(3)}', 17, 17, 'var|xhr|url|null|generateSerial|flag|HTB|flag|new|serial|XMLHttpRequest|send|php|open|POST|true|function'.split('|'), 0, {}))
```

UnPack Clear

```
function generateSerial()  
{  
  ...SNIP...  
  var xhr=new XMLHttpRequest();  
  var url="/serial.php";  
  xhr.open("POST",url,true);  
  xhr.send(null)  
}
```

We can see that this tool does a much better job in deobfuscating the JavaScript code and gave us an output we can understand:

Code: **javascript**


```
function generateSerial() {  
  ...SNIP...  
  var xhr = new XMLHttpRequest;  
  var url = "/serial.php";  
  xhr.open("POST", url, true);  
  xhr.send(null);  
};
```

As previously mentioned, the above-used method of obfuscation is **packing**. Another way of **unpacking** such code is to find the **return** value at the end and use **console.log** to print it instead of executing it.

Reverse Engineering

Though these tools are doing a good job so far in clearing up the code into something we can understand, once the code becomes more obfuscated and encoded, it would become much more difficult for automated tools to clean it up. This is especially true if the code was obfuscated using a custom obfuscation tool.

We would need to manually reverse engineer the code to understand how it was obfuscated and its functionality for such cases. If you are interested in knowing more about advanced JavaScript Deobfuscation and Reverse Engineering, you can check out the [Secure Coding 101](#) module, which should thoroughly cover this topic.



Connect to Pwnbox
Your own web-based Parrot Linux Instance to play our labs.

Pwnbox Location

UK 100ms

[Terminate Pwnbox to switch location](#)

Start Instance

00 / 1 spawns left

Waiting to start

☐ Enable step-by-step solutions for all questions

Questions



Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target(s): [Click here to spawn the target system!](#)

+1

Using what you learned in this section, try to deobfuscate 'secret.js' in order to get the content of the flag. What is the flag?

HTB[1_4m_7h3_53r14l_g3n3r470r!]



Submit



Hint

← Previous

Next →



Mark Complete & Next

