SECURITY+ V4 LAB SERIES

Lab 10: JavaScript Obfuscation & Dead Code Injection

Document Version: 2023-02-27

Material in this Lab Aligns to the Following	
CompTIA Security+ (SY0-601) Exam Objectives	2.3: Summarize secure application development, deployment, and automation concepts
All-In-One CompTIA Security+ Sixth Edition ISBN-13: 978-1260464009 Chapters	11: Secure Application Development, Deployment, and Automation Concepts

Contents

Introduction

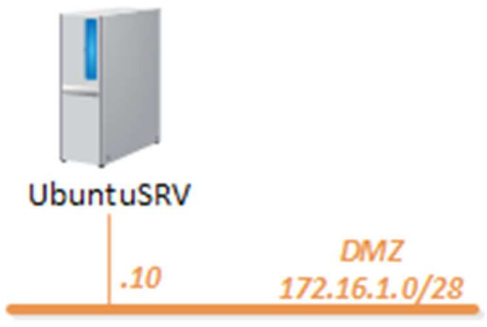
In this lab, you will see how to find and access JavaScript on a local machine and learn the commonly applied techniques for protecting JavaScript.

Objective

In this lab, you will perform the following tasks:

- Perform JavaScript obfuscation
- Perform dead code injection

Lab Topology



Lab Settings

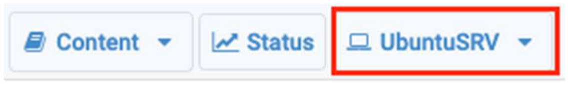
The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	**IP Address**	**Account** (if needed)	**Password** (if needed)
UbuntuSRV	`172.16.1.10`	`sysadmin`	`NDGlabpass123!`

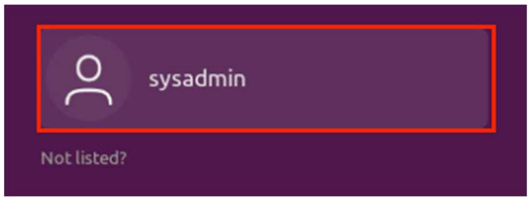
1 JavaScript Obfuscation

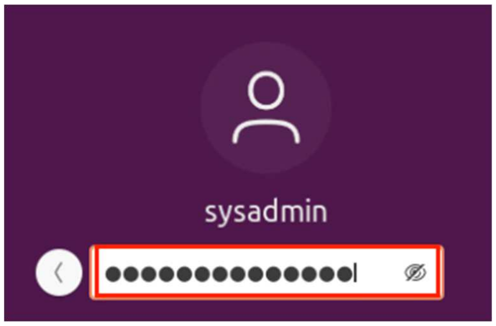
1.1 Obfuscate the nodejs Codes

1. Click on the **UbuntuSRV** tab to access the *UbuntuSRV* VM.

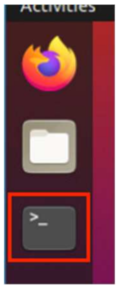


2. Log in to the UbuntuSRV as username `sysadmin`, password `NDGlabpass123!`.

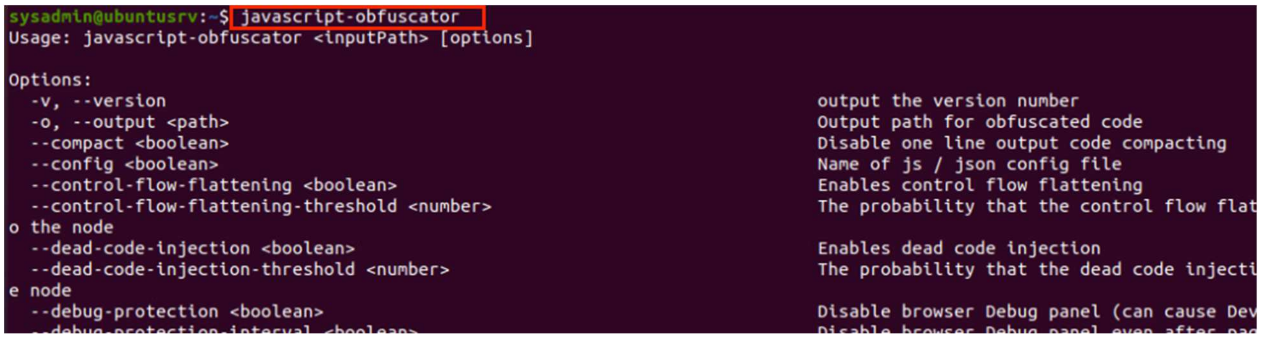




3. Open a *Terminal* window by clicking on the **Terminal** icon located in the left menu pane.

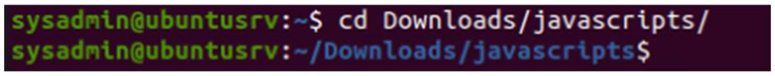


4. In the *Terminal* window, type `javascript-obfuscator` and press **Enter**. Observe the output and learn about how to run the program.



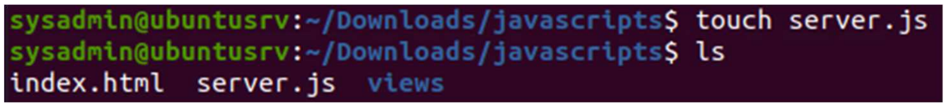
```
sysadmin@ubuntusrv:~$ javascript-obfuscator
Usage: javascript-obfuscator <inputPath> [options]

Options:
  -v, --version                                     output the version number
  -o, --output <path>                               Output path for obfuscated code
  --compact <boolean>                               Disable one line output code compacting
  --config <boolean>                                Name of js / json config file
  --control-flow-flattening <boolean>               Enables control flow flattening
  --control-flow-flattening-threshold <number>      The probability that the control flow flat
o the node
  --dead-code-injection <boolean>                   Enables dead code injection
  --dead-code-injection-threshold <number>          The probability that the dead code injecti
e node
  --debug-protection <boolean>                      Disable browser Debug panel (can cause Dev
```

5. Next, type the `cd Downloads/javascripts` to go to the *javascripts* directory.



```
sysadmin@ubuntusrv:~$ cd Downloads/javascripts/
sysadmin@ubuntusrv:~/Downloads/javascripts$
```

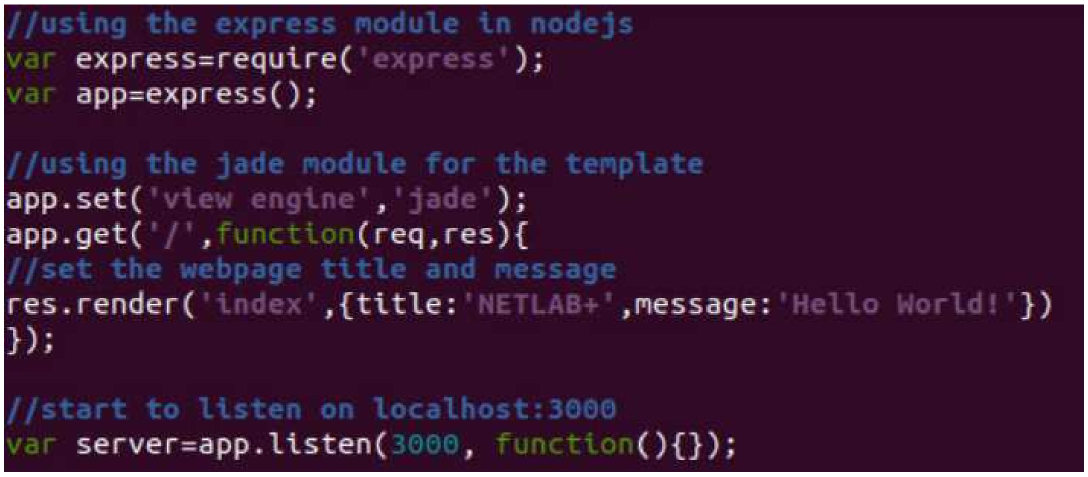
6. We will now create a simple *Hello World!* page using an existing template. In the *terminal* window, type `touch server.js` and press **Enter** to create a file. You can run `ls` to confirm the file was created. The *views* folder stores our template file.



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ touch server.js
sysadmin@ubuntusrv:~/Downloads/javascripts$ ls
index.html  server.js  views
```

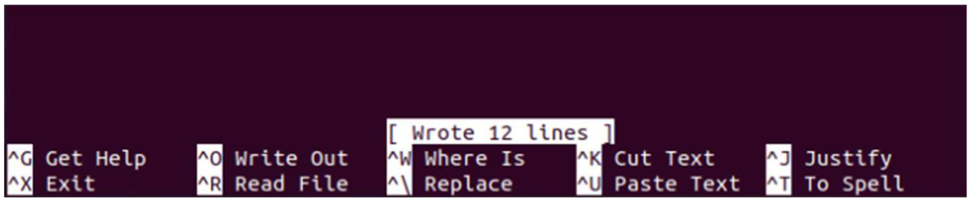
7. Type `nano server.js` and press **Enter.** You will be brought to the nano editor window. Type the following code:

```
//using the express module in nodejs
var express=require('express');
var app=express();

//using the jade module for the template
app.set('view engine','jade');
app.get('/',function(req,res){
//set the webpage title and message
res.render('index',{title:'NETLAB+',message:'Hello World!'})
});

//start to listen on localhost:3000
var server=app.listen(3000, function(){});
```



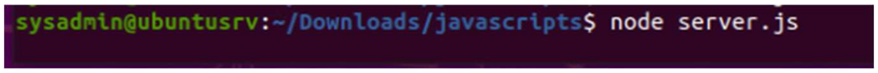
```
//using the express module in nodejs
var express=require('express');
var app=express();

//using the jade module for the template
app.set('view engine','jade');
app.get('/',function(req,res){
//set the webpage title and message
res.render('index',{title:'NETLAB+',message:'Hello World!'})
});

//start to listen on localhost:3000
var server=app.listen(3000, function(){});
```

8. Once finished entering all of the codes, press **Ctrl+S** to save and **Ctrl+X** to quit nano.



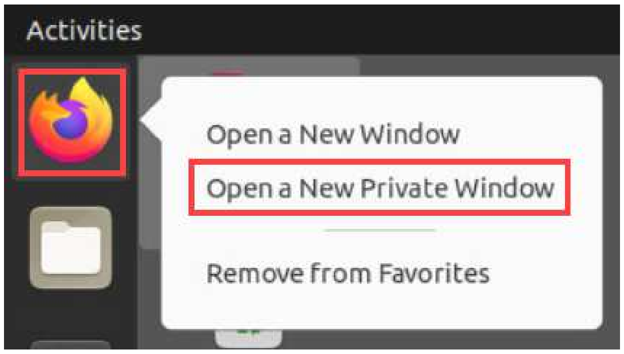
```
                            [ Wrote 12 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell
```

9. Type `node server.js` to bring up the HTTP server.

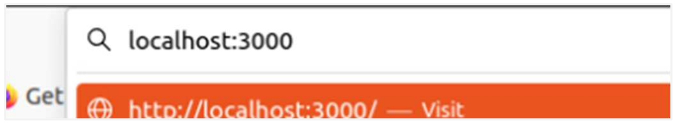


```
sysadmin@ubuntusrv:~/Downloads/javascripts$ node server.js
```

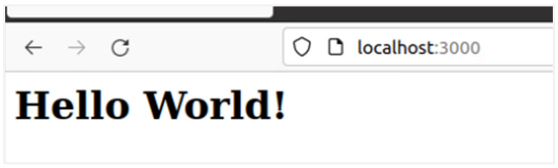
10. Now, let's check our *Hello World!* webpage. On the left side, right-click the **Firefox** icon and choose **Open a New Private Window**.



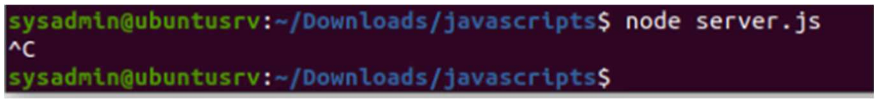
11. In the *Firefox* window, click on the address bar and type `localhost:3000` and then press **Enter.**



12. You should see the *Hello World!* page as shown below:



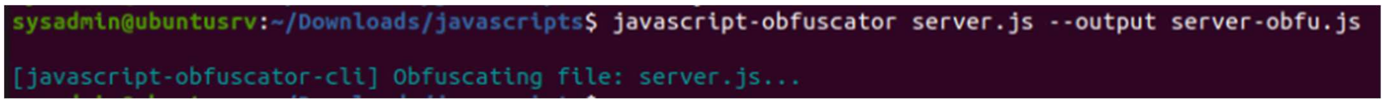
13. Go back to the *Terminal* window, press **Ctrl+C** to stop the server.



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ node server.js
^C
sysadmin@ubuntusrv:~/Downloads/javascripts$
```

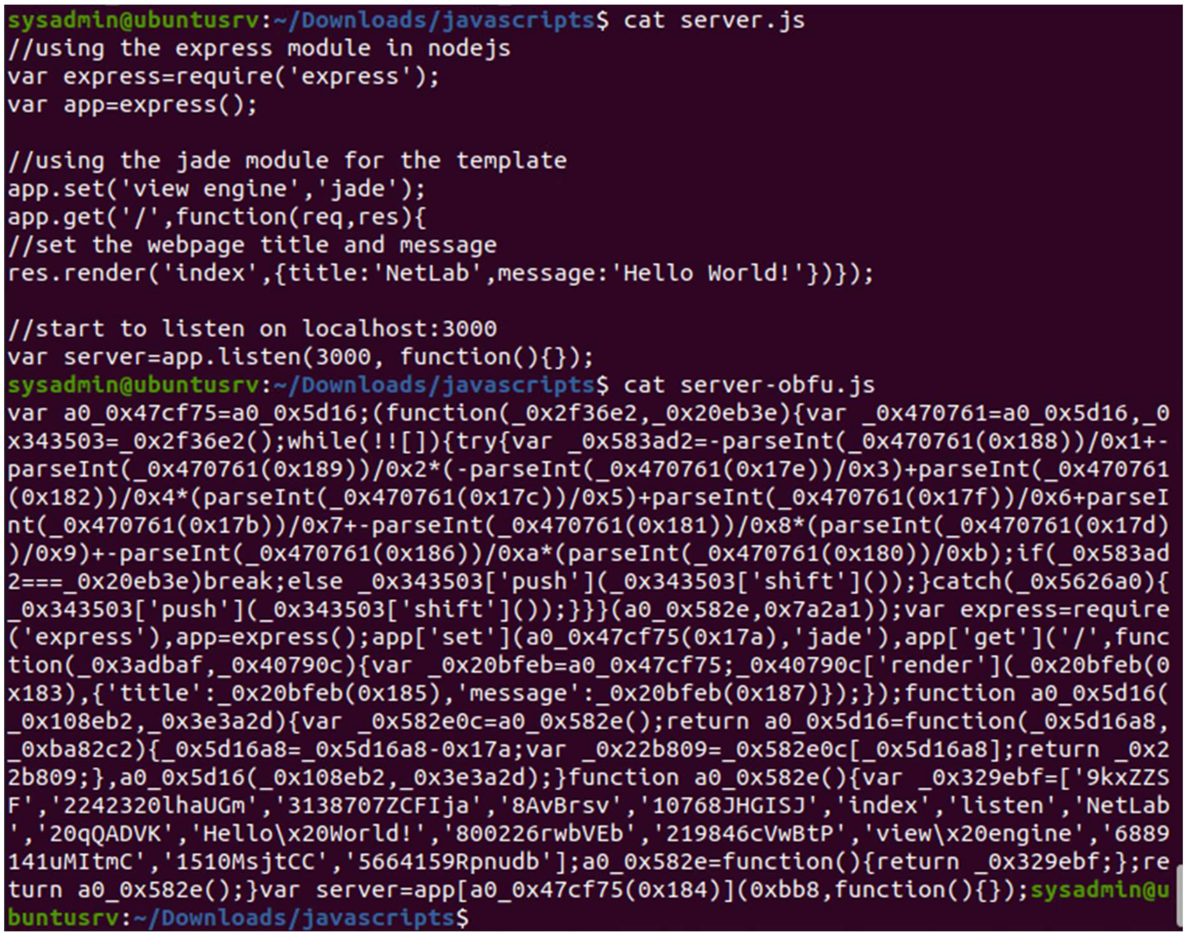
14. Type the following command to obfuscate the JavaScript in the *server.js* file.

```
sysadmin@ubuntusrv:~$ javascript-obfuscator server.js --output server-obfu.js
```



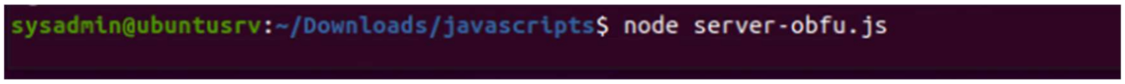
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ javascript-obfuscator server.js --output server-obfu.js

[javascript-obfuscator-cli] Obfuscating file: server.js...
```

15. Now, let's compare the before and after obfuscation. Type `cat server.js` and press **Enter**, then type `cat server-obfu.js` and press **Enter** (your obfuscated code may be different from what is shown in the screenshot below).



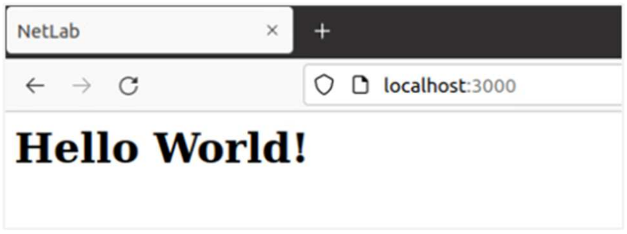
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ cat server.js
//using the express module in nodejs
var express=require('express');
var app=express();

//using the jade module for the template
app.set('view engine','jade');
app.get('/',function(req,res){
//set the webpage title and message
res.render('index',{title:'NetLab',message:'Hello World!'})});

//start to listen on localhost:3000
var server=app.listen(3000, function(){});
sysadmin@ubuntusrv:~/Downloads/javascripts$ cat server-obfu.js
var a0_0x47cf75=a0_0x5d16;(function(_0x2f36e2,_0x20eb3e){var _0x470761=a0_0x5d16,_0
x343503=_0x2f36e2();while(!![]){try{var _0x583ad2=-parseInt(_0x470761(0x188))/0x1+-
parseInt(_0x470761(0x189))/0x2*(-parseInt(_0x470761(0x17e))/0x3)+parseInt(_0x470761
(0x182))/0x4*(parseInt(_0x470761(0x17c))/0x5)+parseInt(_0x470761(0x17f))/0x6+parseI
nt(_0x470761(0x17b))/0x7+-parseInt(_0x470761(0x181))/0x8*(parseInt(_0x470761(0x17d)
)/0x9)+-parseInt(_0x470761(0x186))/0xa*(parseInt(_0x470761(0x180))/0xb);if(_0x583ad
2===_0x20eb3e)break;else _0x343503['push'](_0x343503['shift']());}catch(_0x5626a0){
_0x343503['push'](_0x343503['shift']());}}}(a0_0x582e,0x7a2a1));var express=require
('express'),app=express();app['set'](a0_0x47cf75(0x17a),'jade'),app['get']('/',func
tion(_0x3adbaf,_0x40790c){var _0x20bfeb=a0_0x47cf75;_0x40790c['render'](_0x20bfeb(0
x183),{'title':_0x20bfeb(0x185),'message':_0x20bfeb(0x187)});});function a0_0x5d16(
_0x108eb2,_0x3e3a2d){var _0x582e0c=a0_0x582e();return a0_0x5d16=function(_0x5d16a8,
_0xba82c2){_0x5d16a8=_0x5d16a8-0x17a;var _0x22b809=_0x582e0c[_0x5d16a8];return _0x2
2b809;},a0_0x5d16(_0x108eb2,_0x3e3a2d);}function a0_0x582e(){var _0x329ebf=['9kxZZS
F','2242320lhaUGm','3138707ZCFIja','8AvBrsv','10768JHGISJ','index','listen','NetLab
','20qQADVK','Hello\x20World!','800226rwbVEb','219846cVwBtP','view\x20engine','6889
141uMItmC','1510MsjtCC','5664159Rpnudb'];a0_0x582e=function(){return _0x329ebf;};re
turn a0_0x582e();}var server=app[a0_0x47cf75(0x184)](0xbb8,function(){});sysadmin@u
buntusrv:~/Downloads/javascripts$
```

16. We can see that after the obfuscation, the code became much harder to read and very difficult to revert to its original state. Let's see if the code still runs. Type the command `node server-obfu.js` and then press **Enter**.



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ node server-obfu.js
```

17. We are seeing the same thing as last time; it is almost certain that the HTTP runs just fine. However, to be 100% sure, let's launch a **Firefox Private Window** once again and go to the `localhost:3000` page. It should again show the same *Hello World!* page.



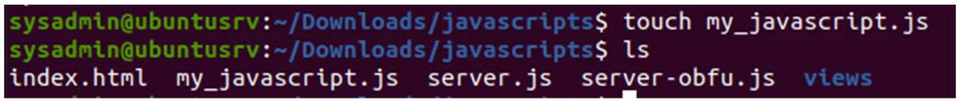
Hello World!

18. Close the *Firefox* window, and go back to your terminal window. Press **Ctrl+C** to end the server.

1.2 Obfuscate JavaScript on an HTML Page

We know that JavaScript will be executed on the client-side; thus, it is critical to protect our JavaScript code from being copied or redistributed by others. Obfuscation is a good way to achieve this goal.

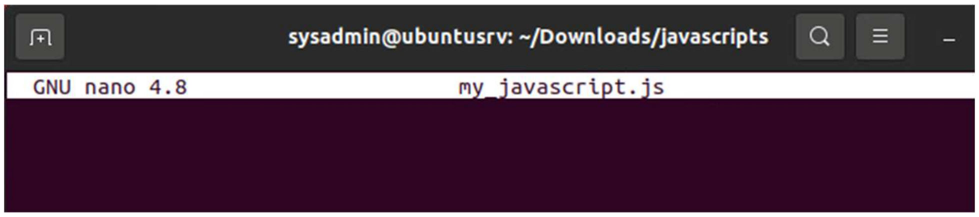
1. Let's write another simple JavaScript function that we will use on an HTML page. Type `touch my_javascript.js` and then press **Enter**. You can check whether the file was created by using the `ls` command.



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ touch my_javascript.js
sysadmin@ubuntusrv:~/Downloads/javascripts$ ls
index.html  my_javascript.js  server.js  server-obfu.js  views
```

The HTML page has been written; you can check the content by using the **cat index.html** command.

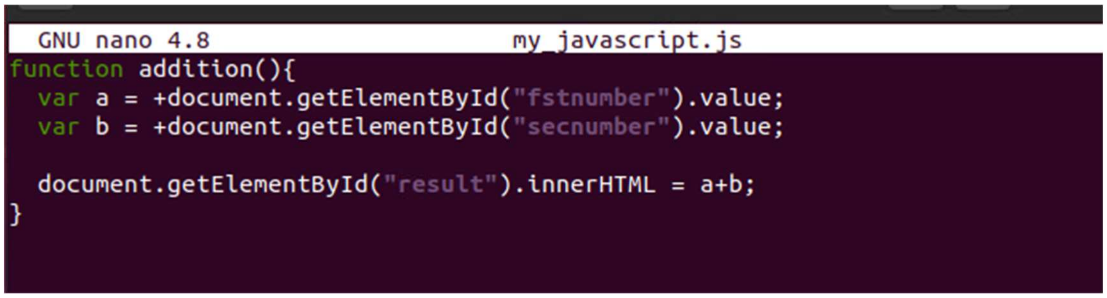
2. Then, we will edit the *my_javascript.js* file content by using the `nano my_javascript.js` command. You will see a screen like this once the nano editor opens.



```
sysadmin@ubuntusrv: ~/Downloads/javascripts
 GNU nano 4.8                    my_javascript.js
```

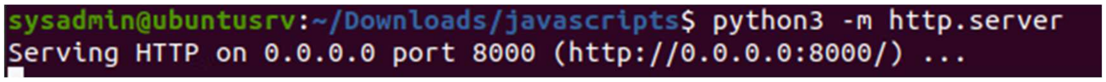
3. Type the following content in the nano editor. Once finished, press **Ctrl+S** to save and **Ctrl+X** to quit the nano editor.

```
function addition() {
  var a = +document.getElementById("fstnumber").value;
  var b = +document.getElementById("secnumber").value;

  document.getElementById("result").innerHTML = a+b;
}
```

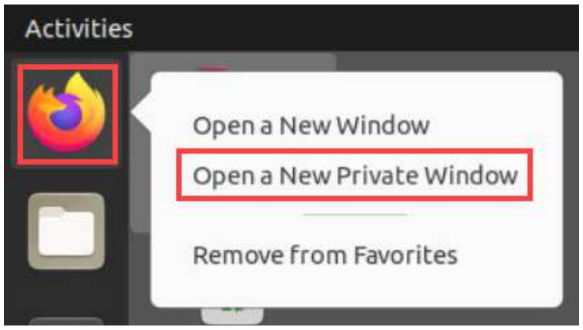


```
  GNU nano 4.8                    my_javascript.js
function addition(){
  var a = +document.getElementById("fstnumber").value;
  var b = +document.getElementById("secnumber").value;

  document.getElementById("result").innerHTML = a+b;
}
```

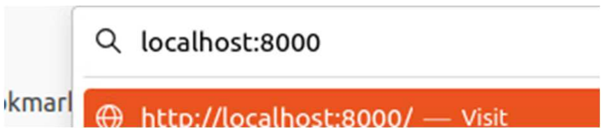
4. Now, let's test our webpage. In the terminal window, type `python3 -m http.server` to start an HTTP server. Open a new **Firefox Private Window**.



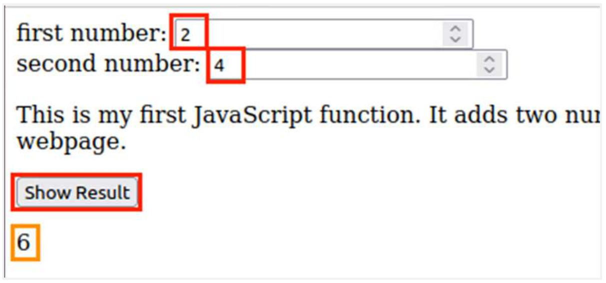
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```



5. In the address bar, go to address `localhost:8000.` We will see our website.



6. We can test our function by entering values for *first number* and *second number*, then click **Show Result**.

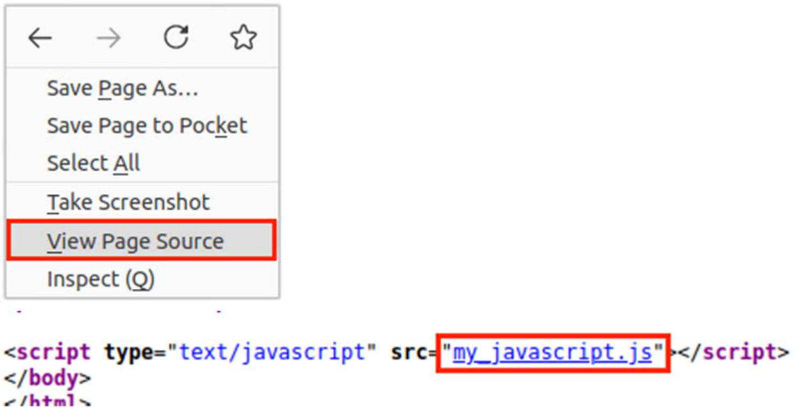


first number: 2
second number: 4

This is my first JavaScript function. It adds two nu
webpage.

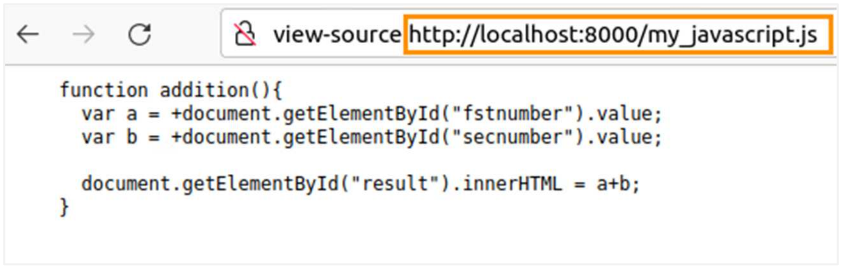
6

7. Right-click on a white space area, then select **View Page Source**. In the new window, click on the **my_javascript.js** link, and we will be taken to our JavaScript code.



```
<script type="text/javascript" src="my_javascript.js"></script>
</body>
</html>
```

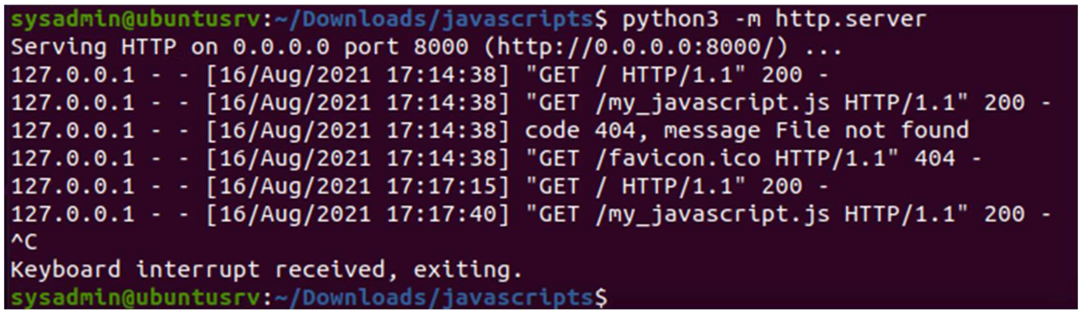
8. Now we can see what we had typed in as our addition function. This is the file that a client could see and would probably redistribute for personal use. For security practices, we can obfuscate this code so it is more difficult for others to directly copy and redistribute. Close the *Firefox* window.



view-source:http://localhost:8000/my_javascript.js

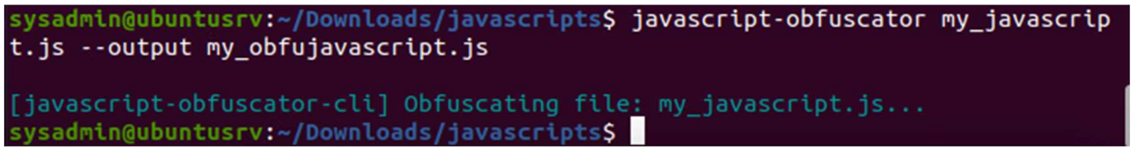
```
function addition(){
  var a = +document.getElementById("fstnumber").value;
  var b = +document.getElementById("secnumber").value;

  document.getElementById("result").innerHTML = a+b;
}
```

9. Switch back to the *Terminal* window and press **Ctrl+C** to exit the HTTP server**.**



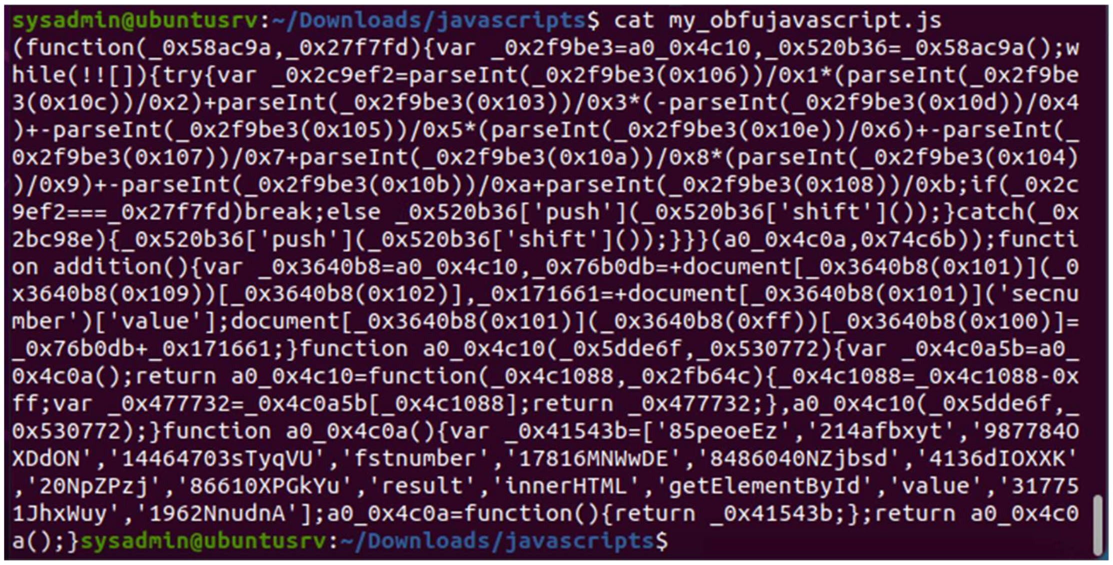
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [16/Aug/2021 17:14:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Aug/2021 17:14:38] "GET /my_javascript.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Aug/2021 17:14:38] code 404, message File not found
127.0.0.1 - - [16/Aug/2021 17:14:38] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [16/Aug/2021 17:17:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Aug/2021 17:17:40] "GET /my_javascript.js HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
sysadmin@ubuntusrv:~/Downloads/javascripts$
```

10. Next, we will use the program again to obfuscate our *my_javascript.js*. Type `javascript-obfuscator my_javascript.js --output my_obfujavascript.js`, then press **Enter.**



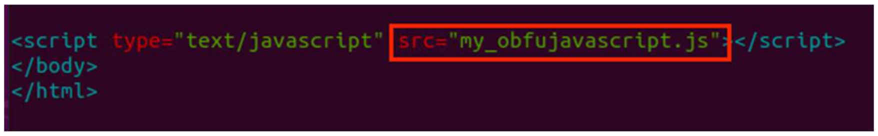
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ javascript-obfuscator my_javascrip
t.js --output my_obfujavascript.js

[javascript-obfuscator-cli] Obfuscating file: my_javascript.js...
sysadmin@ubuntusrv:~/Downloads/javascripts$
```

You can check the obfuscated code by entering the command `cat my_obfujavascript.js`



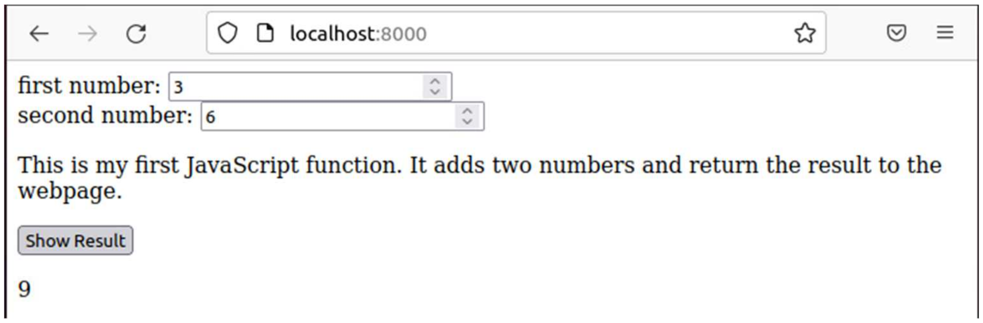
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ cat my_obfujavascript.js
(function(_0x58ac9a,_0x27f7fd){var _0x2f9be3=a0_0x4c10,_0x520b36=_0x58ac9a();w
hile(!![]){try{var _0x2c9ef2=parseInt(_0x2f9be3(0x106))/0x1*(parseInt(_0x2f9be
3(0x10c))/0x2)+parseInt(_0x2f9be3(0x103))/0x3*(-parseInt(_0x2f9be3(0x10d))/0x4
)+-parseInt(_0x2f9be3(0x105))/0x5*(parseInt(_0x2f9be3(0x10e))/0x6)+-parseInt(_
0x2f9be3(0x107))/0x7+parseInt(_0x2f9be3(0x10a))/0x8*(parseInt(_0x2f9be3(0x104)
)/0x9)+-parseInt(_0x2f9be3(0x10b))/0xa+parseInt(_0x2f9be3(0x108))/0xb;if(_0x2c
9ef2===_0x27f7fd)break;else _0x520b36['push'](_0x520b36['shift']());}catch(_0x
2bc98e){_0x520b36['push'](_0x520b36['shift']());}}}(a0_0x4c0a,0x74c6b));functi
on addition(){var _0x3640b8=a0_0x4c10,_0x76b0db=+document[_0x3640b8(0x101)](_0
x3640b8(0x109))[_0x3640b8(0x102)],_0x171661=+document[_0x3640b8(0x101)]('secnu
mber')['value'];document[_0x3640b8(0x101)](_0x3640b8(0xff))[_0x3640b8(0x100)]=
_0x76b0db+_0x171661;}function a0_0x4c10(_0x5dde6f,_0x530772){var _0x4c0a5b=a0_
0x4c0a();return a0_0x4c10=function(_0x4c1088,_0x2fb64c){_0x4c1088=_0x4c1088-0x
ff;var _0x477732=_0x4c0a5b[_0x4c1088];return _0x477732;},a0_0x4c10(_0x5dde6f,_
0x530772);}function a0_0x4c0a(){var _0x41543b=['85peoeEz','214afbxyt','9877840
XDdON','14464703sTyqVU','fstnumber','17816MNWwDE','8486040NZjbsd','4136dIOXXK'
,'20NpZPzj','86610XPGkYu','result','innerHTML','getElementById','value','31775
1JhxWuy','1962NnudnA'];a0_0x4c0a=function(){return _0x41543b;};return a0_0x4c0
a();}sysadmin@ubuntusrv:~/Downloads/javascripts$
```

11. Type `nano index.html`, then press **Enter**. In the nano editor, use the arrow keys on your keyboard to move your cursor to the third line from the bottom, find *my_javascript.js* and change it to `my_obfujavascript.js`. We are going to use our obfuscated JavaScript.



```
<script type="text/javascript" src="my_obfujavascript.js"></script>
</body>
</html>
```

12. Press **Ctrl+S** to save and **Ctrl+X** to exit the nano editor. When you are brought back to the *Terminal* window, run the `python3 -m http.server`, then open a new **Firefox Private Window**. Once more, type the `localhost:8000` to go to our webpage. Enter two random numbers to test whether the function still works.



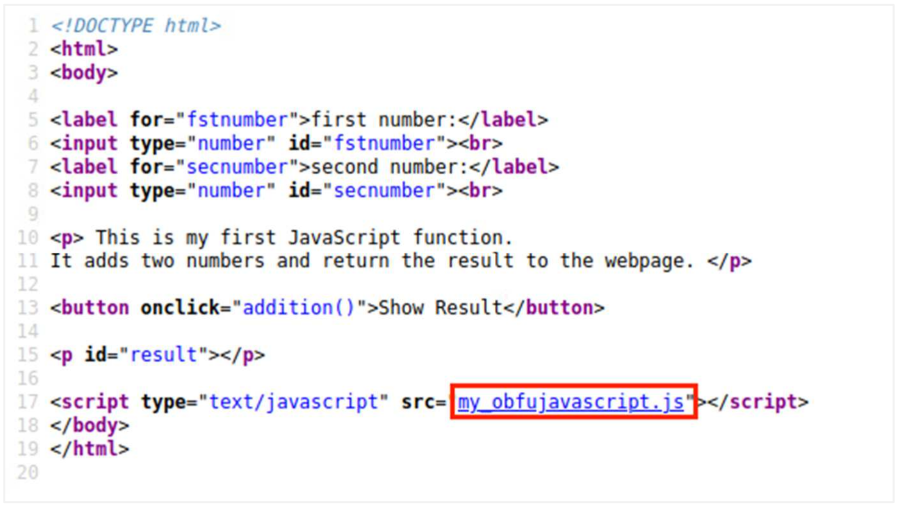
localhost:8000

first number: 3
second number: 6

This is my first JavaScript function. It adds two numbers and return the result to the webpage.

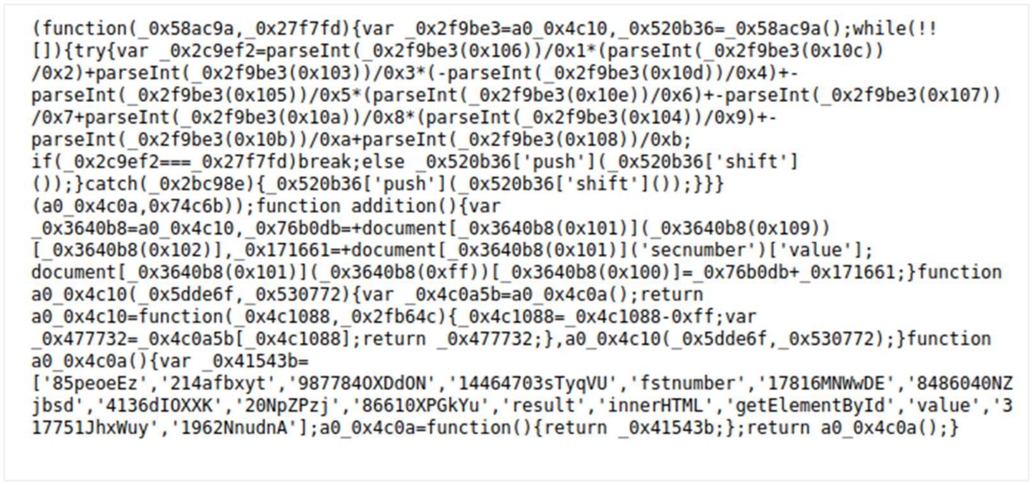
9

13. Right-click on an empty space and choose **View Page Source**. On the new page, it is showing that our obfuscated JavaScript is being used. Click on the **my_obfujavascript.js** link to open the file.

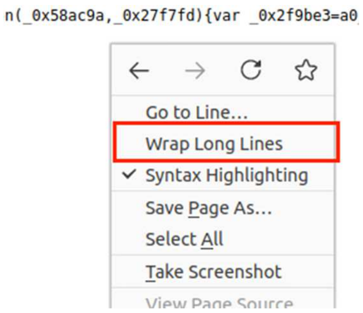


```
<!DOCTYPE html>
<html>
<body>

<label for="fstnumber">first number:</label>
<input type="number" id="fstnumber"><br>
<label for="secnumber">second number:</label>
<input type="number" id="secnumber"><br>

<p> This is my first JavaScript function.
It adds two numbers and return the result to the webpage. </p>

<button onclick="addition()">Show Result</button>

<p id="result"></p>

<script type="text/javascript" src="my_obfujavascript.js"></script>
</body>
</html>
```

14. We can see that the JavaScript code is much harder to read now. If your result shows a long line of code, right-click on an empty space and choose **Wrap Long Lines**.



```
(function(_0x58ac9a,_0x27f7fd){var _0x2f9be3=a0_0x4c10,_0x520b36=_0x58ac9a();while(!!
[]){try{var _0x2c9ef2=parseInt(_0x2f9be3(0x106))/0x1*(parseInt(_0x2f9be3(0x10c))
/0x2)+parseInt(_0x2f9be3(0x103))/0x3*(-parseInt(_0x2f9be3(0x10d))/0x4)+-
parseInt(_0x2f9be3(0x105))/0x5*(parseInt(_0x2f9be3(0x10e))/0x6)+-parseInt(_0x2f9be3(0x107))
/0x7+parseInt(_0x2f9be3(0x10a))/0x8*(parseInt(_0x2f9be3(0x104))/0x9)+-
parseInt(_0x2f9be3(0x10b))/0xa+parseInt(_0x2f9be3(0x108))/0xb;
if(_0x2c9ef2===_0x27f7fd)break;else _0x520b36['push'](_0x520b36['shift']
());}catch(_0x2bc98e){_0x520b36['push'](_0x520b36['shift']());}}}
(a0_0x4c0a,0x74c6b));function addition(){var
_0x3640b8=a0_0x4c10,_0x76b0db=+document[_0x3640b8(0x101)](_0x3640b8(0x109))
[_0x3640b8(0x102)],_0x171661=+document[_0x3640b8(0x101)]('secnumber')['value'];
document[_0x3640b8(0x101)](_0x3640b8(0xff))[_0x3640b8(0x100)]=_0x76b0db+_0x171661;}function
a0_0x4c10(_0x5dde6f,_0x530772){var _0x4c0a5b=a0_0x4c0a();return
a0_0x4c10=function(_0x4c1088,_0x2fb64c){_0x4c1088=_0x4c1088-0xff;var
_0x477732=_0x4c0a5b[_0x4c1088];return _0x477732;},a0_0x4c10(_0x5dde6f,_0x530772);}function
a0_0x4c0a(){var _0x41543b=
['85peoeEz','214afbxyt','9877840XDdON','14464703sTyqVU','fstnumber','17816MNWwDE','8486040NZ
jbsd','4136dIOXXK','20NpZPzj','86610XPGkYu','result','innerHTML','getElementById','value','3
17751JhxWuy','1962NnudnA'];a0_0x4c0a=function(){return _0x41543b;};return a0_0x4c0a();}
```



```
n(_0x58ac9a,_0x27f7fd){var _0x2f9be3=a0
```

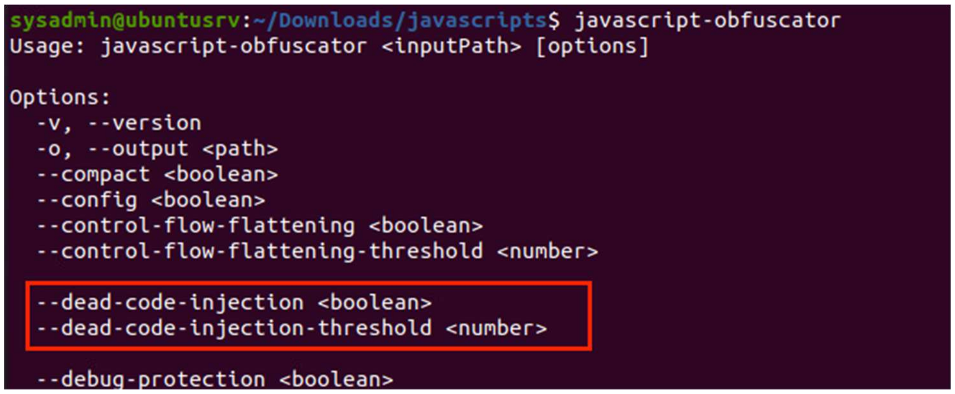
15. Congratulations, you now know how to obfuscate to protect your intellectual property. Close *Firefox* and press **Ctrl+C** in the terminal window to stop the HTTP server. Leave the *Terminal* window open and proceed to the next task.

2 Dead Code Injection

Dead code means there is a part of a program that does nothing when executed. It may create variables and perform mathematical calculations, but the result will never be used for normal operations. Dead code will introduce calculation overhead, so please make sure you select a proper amount of dead code to inject. You do not want to exhaust the CPU before the execution of an important function.

2.1 Dead Code Injection

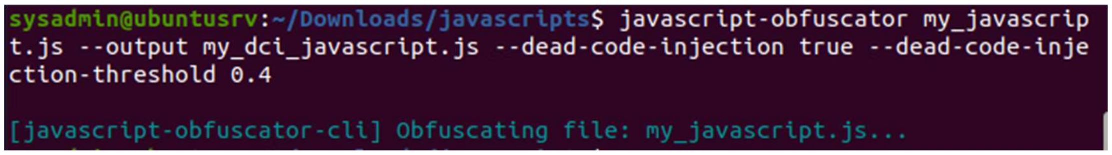
1. While the *Terminal* window remains open, let's click on it. Type `javascript-obfuscator` to observe the options.



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ javascript-obfuscator
Usage: javascript-obfuscator <inputPath> [options]

Options:
  -v, --version
  -o, --output <path>
  --compact <boolean>
  --config <boolean>
  --control-flow-flattening <boolean>
  --control-flow-flattening-threshold <number>

  --dead-code-injection <boolean>
  --dead-code-injection-threshold <number>

  --debug-protection <boolean>
```

2. We can see that the program takes two options related to the dead code injection: dead-code-injection <Boolean> and dead-code-injection-threshold <number>. Here is an example of how to use them:

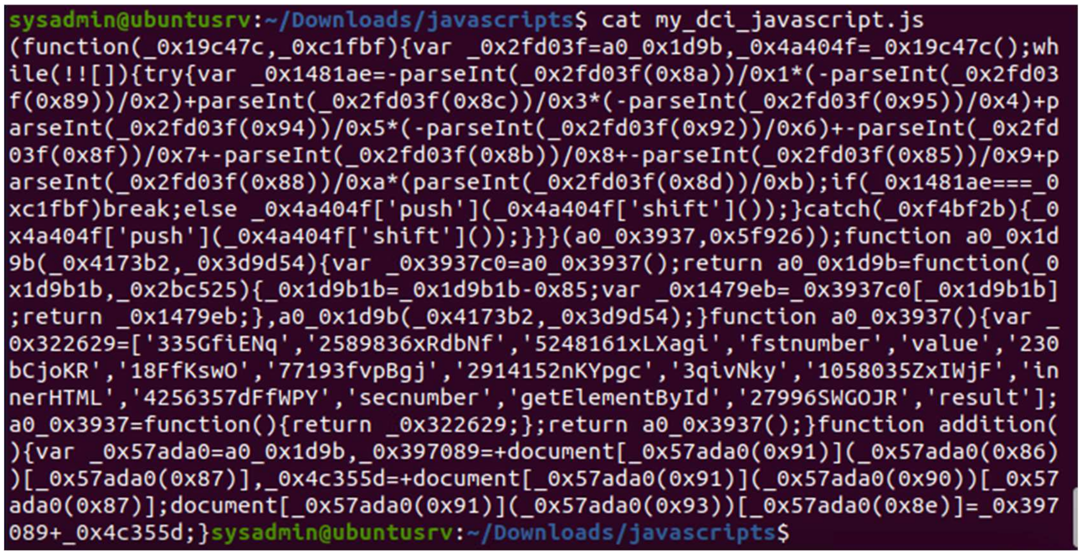
```
javascript-obfuscator my_javascript.js --output my_dci_javascript.js --dead-code-injection true --dead-code-injection-threshold 0.4
```



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ javascript-obfuscator my_javascrip
t.js --output my_dci_javascript.js --dead-code-injection true --dead-code-inje
ction-threshold 0.4

[javascript-obfuscator-cli] Obfuscating file: my_javascript.js...
```

dead-code-injection <Boolean> option is a switch that tells the program whether to add dead code.
dead-code-injection-threshold <number> option sets the possibility a node will be injected with dead code. default is set to 0.4, maximum is 1 and minimum is 0.

3. To observe the code after the injection, type `cat my_dci_javascript.js`, then press **Enter.**



```
sysadmin@ubuntusrv:~/Downloads/javascripts$ cat my_dci_javascript.js
(function(_0x19c47c,_0xc1fbf){var _0x2fd03f=a0_0x1d9b,_0x4a404f=_0x19c47c();wh
ile(!![]){try{var _0x1481ae=-parseInt(_0x2fd03f(0x8a))/0x1*(-parseInt(_0x2fd03
f(0x89))/0x2)+parseInt(_0x2fd03f(0x8c))/0x3*(-parseInt(_0x2fd03f(0x95))/0x4)+p
arseInt(_0x2fd03f(0x94))/0x5*(-parseInt(_0x2fd03f(0x92))/0x6)+-parseInt(_0x2fd
03f(0x8f))/0x7+-parseInt(_0x2fd03f(0x8b))/0x8+-parseInt(_0x2fd03f(0x85))/0x9+p
arseInt(_0x2fd03f(0x88))/0xa*(parseInt(_0x2fd03f(0x8d))/0xb);if(_0x1481ae===_0
xc1fbf)break;else _0x4a404f['push'](_0x4a404f['shift']());}catch(_0xf4bf2b){_0
x4a404f['push'](_0x4a404f['shift']());}}}(a0_0x3937,0x5f926));function a0_0x1d
9b(_0x4173b2,_0x3d9d54){var _0x3937c0=a0_0x3937();return a0_0x1d9b=function(_0
x1d9b1b,_0x2bc525){_0x1d9b1b=_0x1d9b1b-0x85;var _0x1479eb=_0x3937c0[_0x1d9b1b]
;return _0x1479eb;},a0_0x1d9b(_0x4173b2,_0x3d9d54);}function a0_0x3937(){var _
0x322629=['335GfiENq','2589836xRdbNf','5248161xLXagi','fstnumber','value','230
bCjoKR','18FfKswO','77193fvpBgj','2914152nKYpgc','3qivNky','1058035ZxIWjF','in
nerHTML','4256357dFfWPY','secnumber','getElementById','27996SWGOJR','result'];
a0_0x3937=function(){return _0x322629;};return a0_0x3937();}function addition(
){var _0x57ada0=a0_0x1d9b,_0x397089=+document[_0x57ada0(0x91)](_0x57ada0(0x86)
)[_0x57ada0(0x87)],_0x4c355d=+document[_0x57ada0(0x91)](_0x57ada0(0x90))[_0x57
ada0(0x87)];document[_0x57ada0(0x91)](_0x57ada0(0x93))[_0x57ada0(0x8e)]=_0x397
089+_0x4c355d;}sysadmin@ubuntusrv:~/Downloads/javascripts$
```

2.2 Other Dead Code Injection Options

1. There are other important options you may want to test in the lab setting.

   a. **debug-protection <boolean>** option: when this option is set **true**, the browser, if it has debug mode turned on, may freeze or be non-responsive.

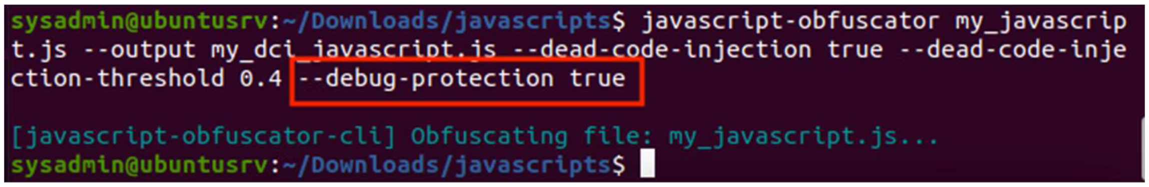
   b. **debug-protection-interval <boolean>** option: when this option is set **true**, the browser may freeze or be non-responsive even after the page is loaded.

   c. **disable-console-output <boolean>** option: when this option is set **true**, the browser console will no longer output console.log, console.info, console.error, and console.warn messages.

   d. **domain-lock** <list> option: when setting this option, the user could provide a list of domain names that can execute the obfuscated code. Any other domain that is not on the list will not be able to execute the obfuscated code.

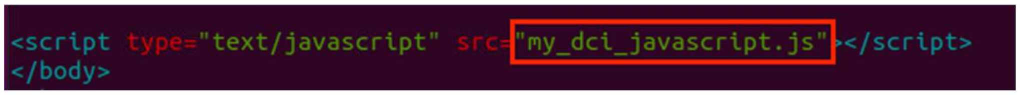
2. To test the above-mentioned options, simply add them at the end of the command. For example, if we want to use the *debug-protection* option, run the command like so:

```
javascript-obfuscator my_javascript.js --output my_dci_javascript.js --dead-code-
injection true --dead-code-injection-threshold 0.4 --debug-protection true
```



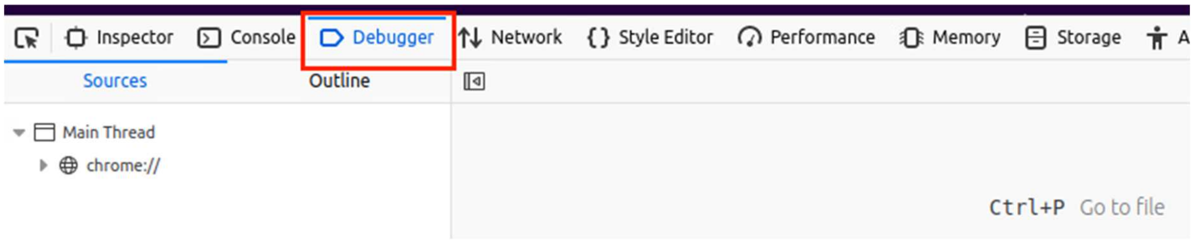
```
sysadmin@ubuntusrv:~/Downloads/javascripts$ javascript-obfuscator my_javascrip
t.js --output my_dci_javascript.js --dead-code-injection true --dead-code-inje
ction-threshold 0.4 --debug-protection true

[javascript-obfuscator-cli] Obfuscating file: my_javascript.js...
sysadmin@ubuntusrv:~/Downloads/javascripts$ 
```

3. Then, edit the *index.html* to reference the *my_dci_javascript.js* file.



```
<script type="text/javascript" src="my_dci_javascript.js"></script>
</body>
```

4. Open a new **Firefox Private Window**, and press **Ctrl + Shift + I** to turn on the *Developer* tools. In the *Developer Tools* area, click to go to the **Debugger** mode.



5. In the address bar, go to `localhost:8000.` You should experience a slow to non-responsive webpage.
6. The lab is now complete; you may end the reservation.