沙盒下的漏洞修补和中间人

吴一昊 201392428

**项目地址：**

**背景**

上次xdctf比赛过程中，全场各队的靶机上，其中一个溢出漏洞均一直在掉分，只有北理的天枢一直守护成功。

各个学校在赛后的交流中，发现各自均因为在漏洞补丁中，触犯了主办方对程序基本逻辑的检测，因此不被判为修复成功，因此即使不再被攻击，仍然被扣分。

主办方后来提醒，漏洞的修补，采用钩子的办法是合法的。这意味着，对于一类漏洞，有了通用临时修补方案，可以在不了解溢出漏洞细节的情况下，尽快打上一个临时补丁。

思考

钩子的编写不是一件容易的事情，难于漏洞利用代码的编写，因此守分并不比夺分容易。因此实现一个通用的临时漏洞补丁方案，是非常重要的。

利用调试器原理，写一个沙盒工具，让含有漏洞的程序跑在沙盒中。监视指定的API或函数，达到中间人的效果。

**初步实现**

利用了一下午的时间，安全人员最喜爱的语言之一python，实现了两个demo。利用了方便强大的调试引擎pydbg，这个引擎为我们封装了内存断点的内部实现等等繁琐细节，另外用到了PE解析工具pefile，方便查找程序入口点，基址等信息。

1、 利用钩子挂在指定API上，可以是装载时链接的，也可以是运行时链接的。通过从函数入口处向后寻找retn n指令，直接jmp到此处，实现API的bypass。

2、 利用钩子挂在任意地址上，修改栈上的内容，达到中间人攻击的效果。

这两个demo我都使用曾经遇到过的一个破解程序的题目，来验证攻击效果。该题目是被加上了密码验证的中国菜刀，我们要想办法绕过验证。

利用第一个demo，通过对ExitProcess的bypass，成功地绕过密码验证，进入到菜刀界面。

利用第二个demo，挂到输入的密码的检测函数上，我们可以修改该函数的参数为任意字符串，相当于对该图形界面的操作界面，提供了一个命令行终端，现在我们便可以写一个脚本来暴力破解该程序的密码了。

再回到本来的目的上来，由于靶机攻击成功与否，是通过socket传输的一个随机长字符串（被称为flag）来判断的，因此即使攻击成功，只要我们对socket api做一定手脚，那么对方仍然不能夺走自己的分数。

**不足之处**

1. 显然demo程序不具有任何扩展性

2. 断点没有次数， 导致通过验证之后，程序不能正常退出，因为所有ExitProcess都被bypass了。

3. 不友好，不能够实时监视我们的攻击行为，不能以人可读的形式显示

4. 不具有可扩展性，demo完全依赖于win32

5. 不能同时完成几个函数或API的同时修改

6. 忽略了大部分代码可能出现的异常

**改进**

基于以上原因，利用两天时间，完成了两个demo的合并，合并为sandbox类，实现了命令行参数、多个处理功能、API名称和地址的通用化、输入数据的可读化、文档完善、日志输出等。

仍然有很多问题：

跨平台依然没有完成，这是一个大问题，因为服务器linux比较多，另外比赛靶机也以linux为主。

程序实现了多个处理功能，但是命令行参数只能输入一组

等等

**后续计划**

交互性优化

完善日志

调用轨迹实现，通过正常代码和poc代码的源代码覆盖模式的不同比较，来研究漏洞原理，实现快速的重放攻击的代码实现。

模糊测试

蜜罐

**攻击展示和usage：**

