Hook可以用系统的setwindowshook或者自己写apihook

dll注入到 模块中 隐藏外挂模块 接收用户的指令 交互通道

安全保护

隐藏的进程

1、注入注册表 设置键值

2、远线程注入 开启远线程 加载模块CreateRemoteThread LoadLiberary

-把dll路径写入进程 开启远线程 参数为LoadLiberay 和其地址 （容易防御）

3、借用可信任进程 远线程 dll注入到service进程 service调用函数注入到目标进程

卸载service中的dll 遍历进程

4、APC注入 异步过程调用 挂起时 线程插入一个APC QuenueUserAPC

5、消息钩子注入

6、输入表注入

7、创建进程时注入 悬挂 双进程保护

8、LSP注入

9、输入法注入

10、comres注入、

三、隐藏模块

抹掉pe头 断开ldr\_moudle链

四、交互通道 传递输入给dll 消息钩子 hook原有的消息处理过程 获取键状态 进程间通信

五、 call函数 外挂call 防止被检测

1、栈帧检测 检测调用处的返回地址 确定是否非法调用

获取函数地址 getmoudlehandle getprocadrress 将参数入栈然后call

在被调用函数中检测调用 地址 hook ce远线程注入

绕过 设定自定义函数头 jmp到函数后面

多层堆栈检测 函数中调用的函数 调用链 多层堆栈回溯检测 返回地址是否非法

制造假栈桢，避免回溯

如何定位call，找call 差异思想 虚函数差异调用 send()回溯

多态是虚函数实现，功能函数以虚函数存在 对象和虚表

六、Hook大全 内联hook和非内联hook

Inline hook 其实就是改写原代码成跳转 需要先把原代码保存 jmp 或者push ret

修改地方可以在系统dll函数调用中

2、输入表hook 开辟空间注入自己的函数修改iat 让函数的地址转到自己的函数

3、虚表hook 类 虚函数 虚表:一个数组，存放虚函数地址 rdata

对象内存布局和虚表

Riit运行时类型识别 结构体 可以解析出来

Hook虚表 call[ 地址长度不好认 应该监控虚表调用 用假虚表 较大 跳转表 监控函数

4、detours hook 函数库 目标函数 跳板函数 截获调用之后执行的函数

事务机制

5、硬件断点hook seh veh tib-fs:[0]异常注册

检测 通过检测线程环境

七。、应用层防护

1、去除静态特征 字符串 全局变量 常量 关于交叉引用

如何得到调用 引用

反dump 修改映像大小 pe头等 改变页面访问性

6、

八、分析工具 GameSpider dll模块注入到进程中 方便调试

Kernel detective

1. 定位保护模块 ring0(驱动) kd 驱动列表 比较加载前后的不同
2. Ring3 回溯法 堆栈调用回溯

差异法 检查dll 防止被发现可以强行挂起其他线程

Dll 文件和内存的比较 一些dll中的调试函数可能被hook掉了，导致调试不能正常运行

如远线程断点函数

1. 自加载 模拟加载器自行加载了系统dll 不用原生接口，避免hook原生系统dll

各模块的导出函数

4、保护方案 hook ntdll.dll的 远线程断点函数 和断点函数 以及页面保护修改函数

Veh无需修改代码

Ring0 ssdt hook ssdt shadow hook

防御性hook的方案 防御内存读写 防御打开挂接进程 防御挂起线程

防御硬件断点 检测线程环境 驱动通信hook 防止时钟加速 防止关闭对象

九、射击游戏

1、模拟按键 ring3 发送消息 sendInput ring0可以用KeyboardClassServiceCallback

或者直接通过out/in操作物理端口

取坐标 ，取颜色 ，模糊找色

2、压枪 平衡Y轴

3、direct hack com编程 hack渲染方式setRenderState 线框透明 禁止光照 禁用烟雾 alpah

D3d9hook endscene drawprimitive drawindexdprimitive

十、外挂检测

代码篡改 可以通过修改检测代码来过 dump后比较

非法call 监控调用 栈桢检测

数据篡改 线程转移 消息分流

主线程挂起 则无法接受任何消息 把主线程移到分线程