# 一种基于内核事件的Windows系统游戏反外挂方法

傅建明* 杨 铮 罗陈可 黄坚伟

(武汉大学国家网络安全学院 空天信息安全与可信计算教育部重点实验室 武汉 430072)

**摘 要**：针对目前客户端反外挂方法的诸多局限，该文提出一种基于内核事件的网络游戏反外挂方法，并实现了反外挂系统CheatBlocker。该方法通过监控Windows系统中的内核事件监视和拦截进程间的异常访问及异常模块注入，同时从内核注入反外挂动态加载库(DLL)用以阻断鼠标键盘的模拟。实验结果表明，CheatBlocker可防御进程模块注入外挂和用户输入模拟类外挂，且具有较低的性能开销。而且，CheatBlocker无需修改内核数据或代码，相比于目前的反外挂系统具有更好的通用性与兼容性。

**关键词**：游戏外挂；反外挂；模块注入；内核事件

**中图分类号**：TN918; TP309 **文献标识码**：A **文章编号**：1009-5896(2020)09-2117-09

**DOI**: 10.11999/JEIT190695

# An Anti-cheat Method of Game Based on Windows Kernel Events

FU Jianming YANG Zheng LUO Chenke HUANG Jianwei

(*Key Laboratory of Aerospace Information Security and Trusted Computing of Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan* 430072, *China*)

**Abstract**: In view of many limitations of current client anti plug-in methods, an anti-cheat method based on kernel events is proposed, and the network game anti-cheat system called CheatBlocker is implemented. This method uses the kernel event monitoring provided by Windows to intercept the abnormal access between processes and the injection of abnormal modules. At the same time, the anti-cheat Dynamic Loaded Library (DLL) injected from the kernel can block the simulation of the mouse keyboard. The experimental results show that CheatBlocker can defend against process module injection cheating and user input simulation cheating, and has low performance overhead. Moreover, CheatBlocker does not need to modify the kernel data or code which ensures the integrity of the kernel and is more compatible than the current anti-cheat systems.

**Key words**: Game cheating; Anti-cheating; Module injection; Kernel event

## 1 引言

网络游戏作为一种娱乐项目，以其上手易、休闲性、社交性强等特点受到广大网民的喜爱。据《2018中国游戏产业报告》显示，2018年上半年，中国游戏市场的用户规模达5.3亿人，实际销售收入达到1050.0亿元[1]。在游戏中，不法用户常常借助游戏辅助程序(简称“外挂”)欺骗游戏客户端及服务器来快速获取游戏利益。外挂滋生了私服、盗号、打金工作室、网络信息欺骗等一系列灰色产业[1]，不仅危害了广大玩家的游戏公平性，也严重影响着网络游戏的营收。另外，外挂也会造成玩家自身的财产损失。

收稿日期：2019-09-09；改回日期：2020-06-13；网络出版：2020-07-18
*通信作者： 傅建明 jmfu@whu.edu.cn
基金项目：国家自然科学基金(61972297, U1636107)
Foundation Items: The National Natural Science Foundation of China(61972297, U1636107)

根据网络游戏客户端/服务器的架构特性，游戏外挂主要通过篡改游戏进程数据、模拟用户输入以及篡改游戏与服务器的通信封包等方式[2]实现外挂功能。篡改游戏进程即通过逆向分析、反汇编等技术分析游戏内部函数调用逻辑和关键数据存储等[3]，静态或动态地修改可执行文件或进程，改变程序执行流程以及提取和修改敏感数据[4]；模拟用户输入即通过程序来模拟用户的鼠标或键盘输入行为，能极大地提高外挂使用者的操作效率[5]，非常适合频繁按键、鼠标点击等重复性操作；篡改通信封包类似于网络攻击中的中间人攻击，主要通过网络抓包截取游戏客户端与服务器的通信数据包，分析报文格式及内容，定位关键数据项在报文中的位置并对其进行篡改，并将篡改后的数据包发送给服务器或客户端[2]。

上述外挂技术中，篡改游戏进程和模拟用户输入的攻击目标均为游戏客户端，而篡改和伪造通信

封包的攻击目标则是客户端与服务器的交互过程。本文的研究重点为游戏客户端的外挂检测与防御，因此本文不再讨论和服务器相关的外挂技术。

进程篡改类外挂的攻击包括外挂模块的注入以及外挂行为的实施两个阶段。为防御进程模块注入，文献[6]提出由进程自身定期扫描内存或进程进程环境信息块(Process-Environment-Block, PEB)的PsModuleLoadedList 列表来检测自身进程是否载入了异常模块。但是，外挂动态加载库(Dynamic Loaded Library, DLL)通过定位到自身模块在游戏进程中的内存地址，并抹去其PE头部，或将自身从游戏进程的PsLoadedModuleList列表中移除[7]，就能逃逸该方法的检测。文献[8]提出了在游戏运行时动态检测游戏代码完整性并对其进行实时修复，但该方法无法发现游戏进程中数据区段的修改；文献[9]使用了内核层系统函数Hook的方法来限制其他进程对游戏进程的访问，但该方法需要修改内核中的系统服务分发表(System-Service-Dispatch-Table, SSDT)Hook相关内核函数。Windows为了保证内核安全性，在Vista之后的所有64位系统中引入了PatchGuard保护机制[10]，禁止修改关键内核数据。由于SSDT属于PatchGuard的保护对象，因此该防御方法难以应用到64位系统上。

对于用户输入模拟类外挂，游戏进程难以判断键盘鼠标消息事件的来源是真正的外设操作还是外挂的模拟行为[11]。文献[12]提出了利用黑名单检测来识别主流的用户输入模拟类外挂，但其检测效果依赖于黑名单的更新频率。文献[5]提出向系统中所有进程都注入DLL并以应用层Hook的方式监控鼠标键盘模拟类API，阻止其它进程向游戏进程发送鼠标键盘消息，该方法弥补了黑名单的不足。但定期遍历系统进程列表来检测新创建的进程的方法存在TOCTOU(Time-Of-Check-to-Time-Of-Use)的局限性，而且会增加性能开销[13]。除此之外，文献[14]提出了根据用户交互特点进行行为建模以及特征分类的方法来区分真实用户与模拟外挂。但由于对游戏玩家的高精度行为建模难度较大，该方法目前仍存在较大的误报率。

由以上分析可知，在目前的客户端反外挂防御中，通常使用扫描游戏自身进程[6]或内核层Hook[9]的方法来防御模块注入类外挂，使用Hook应用层模拟输入类API[5]的方法来防御模拟用户输入的外挂。但这种扫描或者Hook技术存在TOCTOU局限性，且内核Hook技术会破坏系统的完整性，引发兼容性和安全性问题。针对当前客户端外挂防御的局限性，本文设计并实现了基于Windows内核事件的反外挂系统CheatBlocker。从内核层监控其他进程对游戏进程的访问，防止游戏进程内的代码以及关键数据信息遭到恶意修改；其次，由内核层驱动向系统中所有进程注入反外挂DLL，并通过应用层Hook技术监控鼠标键盘模拟类应用程序接口(Application Programming Interface, API)，阻止外挂对游戏进程进行用户输入模拟。本文的主要贡献如下：

(1) 从篡改游戏进程和模拟用户输入两方面分析了游戏客户端外挂的基本原理，为反外挂检测提供了防御基线。

(2) 提出了一种基于内核事件的Windows系统游戏反外挂方法，该方法几乎消除了外挂攻击的时间窗口，且保证了系统自身的完整性。并以此实现了反外挂系统CheatBlocker。

(3) 实验结果验证了本文反外挂方法的有效性。与已知的反外挂方法相比，本文方法具有更好的防御效果与兼容性。

## 2 客户端外挂原理

针对游戏客户端的外挂，根据其行为特点主要可分为进程篡改类和用户输入模拟类。进程篡改类外挂向游戏进程注入外挂模块(代码)，修改游戏进程；用户输入模拟类外挂则调用Windows提供的相关API，实现鼠标键盘的模拟操作。

### 2.1 篡改游戏进程

进程篡改类外挂一般通过模块注入来实现代码注入和修改，它是该类外挂所使用的核心技术。攻击流程为首先向游戏进程注入外挂DLL，由DLL入口函数来执行其外挂代码，再通过Hook Windows API、设置定时器监测进程状态以及创建子线程等方式实现外挂功能[15]。常见的模块注入方式包括：创建远程线程[15]、插入APC、线程劫持[16]、Hook Windows全局消息[17]和创建恶意输入法[6]。前3种方法均需要修改游戏进程的内存空间，如图1所示。后两种方法则仅利用Windows的系统机制，不涉及与游戏进程的交互，如图2所示。

(1) 插入APC：异步过程调用(Asynchronous Procedure Call, APC)是Windows用来实现异步I/O操作的一种机制。当I/O操作完成时，系统会自动调用该I/O操作关联线程的APC队列上的所有APC函数，使程序通过APC函数来监测I/O操作是否完成。因此，外挂程序可将实现注入功能的函数(例如LoadLibrary)作为APC插入到游戏进程的某一线程的APC队列上。当该线程完成任意I/O操作后，系统会自动调用APC队列上的所有函数，进而将DLL注入到游戏进程中，如图1中攻击A所示。

(2) 线程劫持：该方法首先挂起游戏进程的某

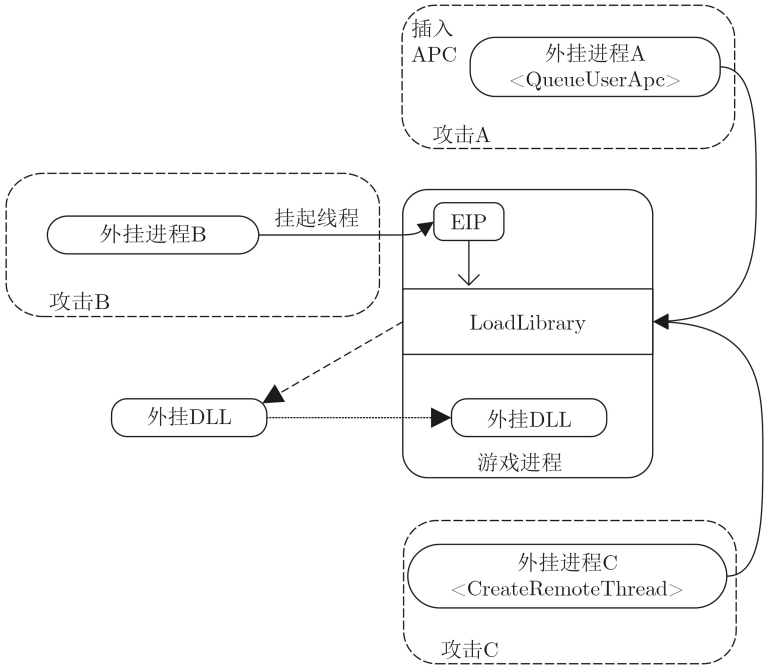
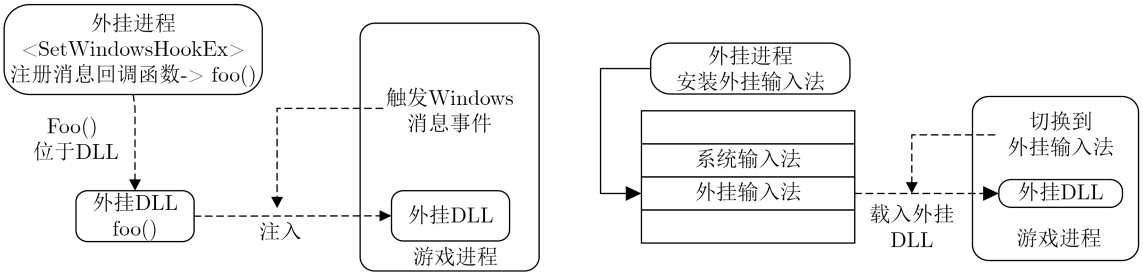



图 1 基于跨进程访问的注入方法




(a) Windows消息注入　　(b) 输入法注入

图 2 基于系统机制的注入方法

一线程，然后修改该线程的指令指针(EIP或RIP)，该指针指向外挂程序在游戏进程中分配的一段内存区域。该区域填写一段代码(简称Shellcode)，该代码实现模块注入。当线程恢复运行后，线程会从修改后的指令处开始运行，即执行Shellcode加载外挂DLL，如图1中攻击 B所示。

(3) 创建远程线程：该方法通过在游戏进程中创建一个远程线程，并利用该线程来完成载入DLL的相关操作。如图1中攻击 C 所示，外挂程序将LoadLibrary的地址写入游戏进程空间，作为远程线程的起始地址，然后将外挂DLL的路径字符串写入游戏进程，并将其地址作为远程线程的参数，然后调用CreateRemoteThread函数创建远程线程，最后远程线程执行并调用LoadLibrary将外挂DLL载入到游戏进程中。

(4) Hook Windows 消息：该方法利用Windows提供的SetWindowsHookEx函数针对某一类消息事件注册一个全局消息钩子，当系统中有线程触发该消息事件时，Windows会自动调用注册时指定的Hook函数。如果该函数位于DLL中，Windows则会首先将DLL注入到触发消息事件的进程中，然后调用Hook函数。如图2(a)所示，外挂程序只要将Hook函数定义在需要注入的DLL中，然后注册消息钩子，当游戏进程触发消息事件时，外挂DLL即被载入到该进程中。

(5) 输入法注入：在Windows平台上，当窗口程序进行输入法切换时，系统会自动将输入法对应的功能文件(Input Method Manager, IME)载入窗口所在的进程中，使其提供相应的输入法功能。如图2(b)所示，外挂程序首先注册一个外挂输入法，并将外挂DLL设定为该输入法的功能文件，然后向用户窗口发送一个输入法切换消息WM_INPUTLANGCHANGE，使窗口将输入法切换至外挂输入法，系统随即自动将外挂DLL载入游戏进程中。

### 2.2 模拟用户输入

用户鼠标模拟类外挂的核心功能是使用独立的程序或脚本模拟用户的输入[18]来代替用户与游戏交互，通常不会向游戏进程注入代码。常见的模拟方法主要分为内核层与用户层模拟两类，内核层模拟主要是加载内核驱动来模拟系统中的键盘设备驱动来向Windows消息队列发送键盘和鼠标等消息[19]。此方法的实现较为底层，不容易被反外挂系统发现，但是需要安装额外驱动程序，普及度不高[19]。为此，外挂制作者通常使用用户层模拟技术[20]。用户层模拟技术主要分为全局模拟和窗口模拟两类[13]，其原理均是使用Windows的消息API在外挂程序中向游戏进程发送鼠标或键盘消息。全局模拟是指应用层程序直接调用这类函数来模拟鼠标或键盘事件。当操作系统底层鼠标或键盘驱动接收到用户交互后(移动鼠标、按键等)，会通过mouse_event和key_bdevent等函数合成一个鼠标或键盘消息，插入到系统消息队列中，然后通知进程获取用户的鼠标或键盘操作。而窗口模拟则主要利用SendMessage, PostMessage等API，该类API向指定窗口发送一个Windows消息，将消息插入到指定窗口所属线程的消息队列上。当发送的消息是鼠标或键盘消息时，就可以在指定窗口中模拟鼠标或键盘操作。

## 3 反外挂系统设计

根据第2节中对相关外挂技术的分析，本文提出了基于Windows内核事件的反外挂系统CheatBlocker，总体架构如图3所示。系统首先通过内核事件监控来阻止外挂对游戏进程进行内存修改或模块注入；其次向系统中所有进程注入反外挂DLL，并通过应用层Hook监控鼠标键盘模拟类API，阻止外挂对游戏进程进行用户输入模拟。

### 3.1 防御进程篡改

由2.1节可知，进程篡改类外挂主要使用模块注入来实现其外挂功能，因此为防御此类外挂，必须阻断其对游戏进程进行模块注入，这也是本文关于进程篡改的防御思路的出发点。由于传统应用层模块注入检测技术存在漏检情况，本文提出了内核层与应用层结合的反注入方法，如图4所示，可以检测到所有的应用层模块注入行为。该方法通过过滤句柄权限来阻止利用跨进程非法访问实现的模块注入，同时利用代码签名来检测利用系统机制的模块注入。

#### 3.1.1 防御跨进程访问注入

第2节介绍了跨进程访问的模块注入方法，例如从源进程向游戏进程中分配内存空间、写入数据

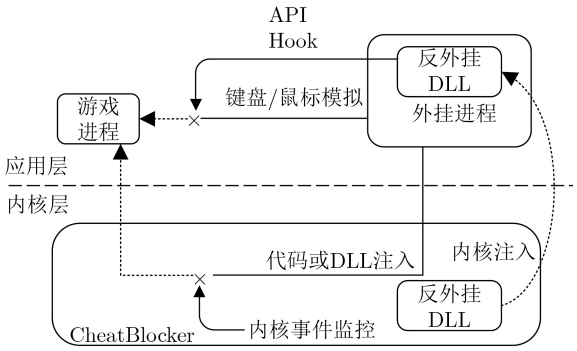


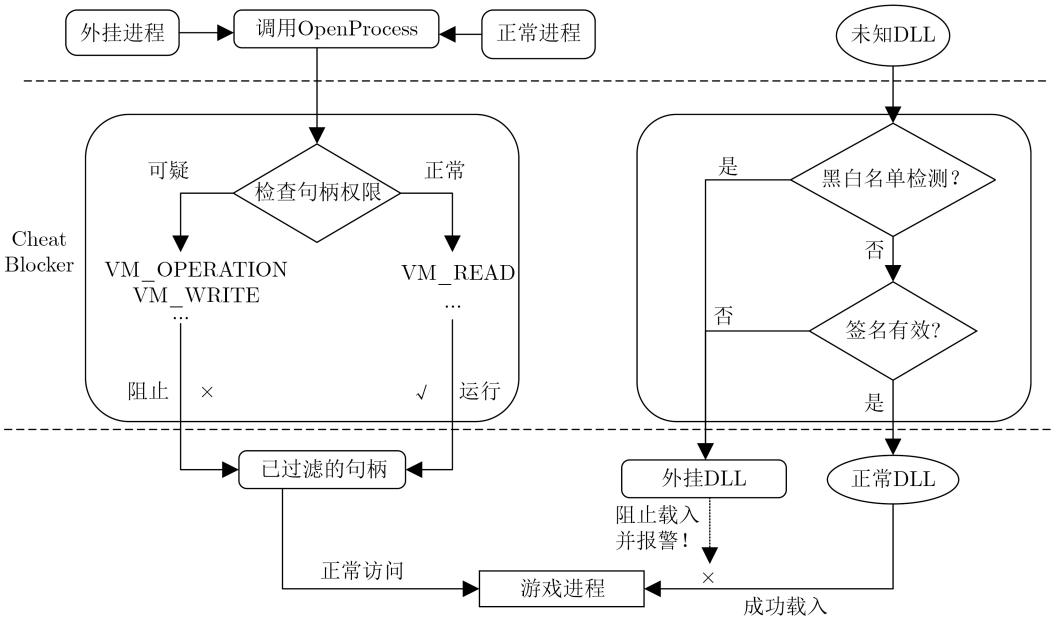

图 3 反外挂系统

以及创建远程线程等。该方法利用的Windows系统函数分别为VirtualAllocEx, WriteProcessMemory和CreateRemoteThread等，这类函数都需要进程句柄作为参数。而且该进程句柄只能通过调用CreateProcess, OpenProcess和DuplicateHandle这3个系统函数访问游戏进程来获取，并由调用者在调用时声明句柄所具有的权限，代表其能够使用该句柄进行哪些跨进程访问操作。同样，其他进程通过跨进程方法访问游戏进程中的某一线程时，需要调用OpenThread等系统函数来获取该线程的句柄，并需指明需要获取的操作权限。

外挂程序对游戏进程进行跨进程访问时，必须获取具有相应权限的进程句柄，如果权限不匹配，则会导致跨进程访问失败，所以检测到句柄的非法创建是防御此类模块注入的关键。本文通过Windows提供的ObRegisterCallback API注册内核对象操作的回调函数，从内核层监视系统中所有进程或线程句柄的创建事件。当其他进程请求创建游戏进程的进程或线程句柄时，系统会自动调用CheatBlocker注册的回调函数，回到函数通过分析所请求的权限是否包含敏感权限来判断该次请求是否非法，如图4(a)所示。如果非法，则直接将不含非法权限的句柄返回给调用者。若调用者使用该句柄进行非法跨进程访问操作，将因为权限不足而导致访问失败，以此实现对此类注入的防御。

#### 3.1.2 防御系统机制注入

通过内核层过滤句柄权限的方式，可使跨进程访问类的注入方法失效，但利用Windows系统机制的注入方法无需获取进程句柄，为了防御此类注入，CheatBlocker采用了内核层监控模块载入事件的方法监控游戏进程中是否载入了非法模块。如图4(b)所示，CheatBlocker通过PsSetLoadImage NotifyRoutine API监控系统中的所有模块载入事件，当其检测到游戏进程内发生模块载入时，首先将载入模块与驱动携带的黑白模块名单进行哈希值匹配，作为第1次过滤；若载入模块不在黑白名单




(a) 监控句柄创建　　(b) 监控DLL载入

图 4 模块注入防御

内，内核监控程序则将模块信息通过加密信道发送给应用层，由应用层进程对模块文件进行代码签名验证。如果该模块没有代码签名或签名损坏及过期，则认为其是可能为外挂模块，并通知游戏进程有非法模块注入。

### 3.2 防御用户输入模拟

利用跨进程访问控制和代码签名能阻止进程篡改，但无法检测用户输入模拟。为此CheatBlocker在文献[5]的研究基础上，在内核层监控新进程创建，并在进程主函数执行前向其注入反外挂DLL，利用该DLL 拦截用户输入模拟API。

对新进程创建事件的监控主要通过监控系统中ntdll.dll模块的载入来实现，因为系统模块ntdll.dll是所有进程第1个载入的模块，此时其他系统DLL和程序自身携带的DLL都还未载入，进程的主程序也未执行，因此该方法能及时注入反外挂DLL，保证外挂进程无法逃逸检测。当检测到ntdll.dll载入时，可认定为有新进程创建，驱动进而通过内存手动映射的方式将反外挂DLL注入到进程中，由其Hook鼠标键盘模拟类的API函数，相关函数如表1所示。由于系统中的所有进程都会被注入该DLL，因而为了保证系统稳定性，不影响正常软件功能，还需对Hook函数进行进一步的过滤操作。

如图5所示，对于需要窗口、进程句柄或进线程id等可以用来标识相关进程的函数(如SendMessage等)，其Hook函数首先会检查此类标识参数和游戏进程有关，且消息类型为鼠标或键盘消息，则使此次调用失败；如果无关或消息类型与鼠标键盘无关，则Hook函数正常调用原始函数。对于不需要标识参数的函数(如SendInput, mouse_event等)，Hook函数首先检查桌面前台窗口是否属于游戏进程。如果是，则可能是外挂程序调用了相关API在游戏进程的窗口内进行模拟操作；如果不是，则认为是正常调用。该方法在文献[5]的基础上增加了对新创建进程的监控，提高了防御效果，且无需定时遍历系统进程列表，降低了系统开销。

**表 1　反外挂DLL Hook函数**

| 模拟类型 | 相关API | API 描述 |
|---|---|---|
| WindowSimulation | SendMessage | 直接向指定窗口发送消息 |
| | PostMessage | 将消息至于指定窗口的消息队列上 |
| | RtlUserSendMessage | SendMessage内部调用API |
| | RtlUserPostMessage | PostMessage内部调用API |
| GlobalSimulation | SendInput | 直接模拟鼠标或键盘操作 |
| | mouse_event | 模拟鼠标 |
| | keyboard_event | 模拟键盘 |

## 4 实验

本文实验平台为两台Windows虚拟机，具体操作系统环境配置如表2所示。本文选择了2款游戏进行实验，第1款为网络游戏《FIFA 10 online》，该款游戏无任何外挂防御行为，针对该游戏的外挂也基本属于进程篡改类。第2款为《穿越火线》(Cross Fire)，所选测试版本为《穿越火线单机版V.1.04》，该版本是《穿越火线》的单机版本，其核心游戏功能和网络版相同，但自身不包含任何反外挂系统或外挂检测模块，因此可作为很好的测试对象。为了验证反外挂系统的功能，本文还在知名外挂制作网站MrAntiFun.net 和网络游戏逆向分析网站unknowncheats.me上收集了多个《FIFA 10》和《穿越火线》游戏外挂以及通用游戏修改工具，相应外挂名称以及其利用的核心外挂技术如表3所示。

### 4.1 功能测试

本文分别在VM1与VM2中对本文提出的反外挂系统CheatBlocker的反外挂功能进行测试。由于CheatBlocker机制为独立于游戏进程运行，并对游戏进程提供实时外挂防护，因此无需对游戏软件进行任何修改，只需将游戏进程添加至CheatBlocker的防护列表中即可。首先在不开启CheatBlocker情况下运行游戏，并使用上述外挂工具逐个进行测试，记录每个外挂是否能够完成相应的外挂功能，如修改游戏内存或注入DLL等。然后在开启CheatBlocker的情况下，重新运行游戏，并再次测试上述外挂能否正常工作并记录。实验结果表明，在32位及64位系统下，若CheatBlocker未开启，则篡改游戏内存和模拟用户输入类外挂均可以正常对游戏进程进行注入或进行用户输入模拟，但在CheatBlocker开启后所有外挂均无法正常工作。

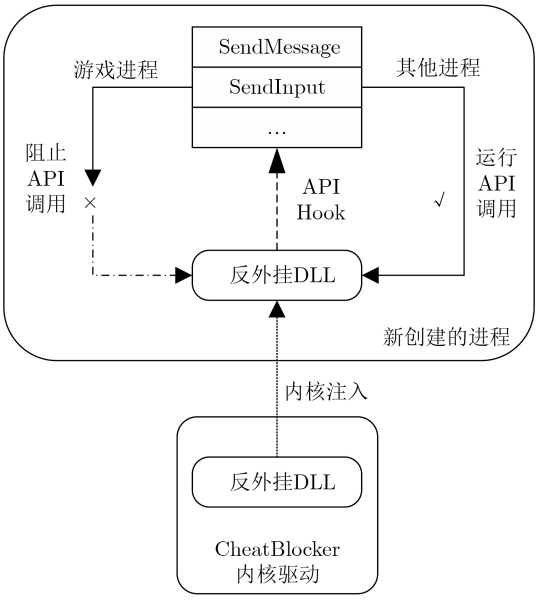



图 5 防御用户输入模拟

**表 2 实验环境**

| VM | CPU | 内存 | 操作系统 |
|---|---|---|---|
| VM1 | 2 cores | 1 GB | Win7 SP1 (64 bit) |
| VM2 | 2 cores | 1 GB | Win7 SP1 (32 bit) |

本文还针对各项外挂技术将CheatBlocker与目前已有的反外挂系统进行了对比实验。对比对象包括Nprotect, Xray, Warden, GameGuard以及EasyAntiCheat。其中前3种均在用户层进行保护；后两者为内核层反外挂系统。对比对象与CheatBlocker同为独立型反外挂系统，可独立于游戏进程运行。本文在反外挂系统开启的情况下，依次使用第2节所述的5种模块注入技术和两种用户输入模拟技术对游戏进程进行测试，即能否向游戏进程中注入DLL或进行鼠标键盘模拟，并记录反外挂系统是否能防御相应的外挂技术。此外，本文还测试了反外挂系统是否支持Windows64位操作系统。各反外挂系统对外挂技术防御情况以及系统兼容性情况如表4所示。可见，CheatBlocker可防御前文所列的所有外挂技术，且支持64位操作系统。而Nprotect, Xray, Warden三者只能防御部分模块注入技术，并且均无法防御用户输入模拟。

在内核层反外挂系统中，GameGuard与CheatBlocker有相似的防御效果，其核心功能主要通过修改Windows系统内核数据以及Hook内核函数实现。其防御思路与CheatBlocker基本相同，但由于内核Hook破坏了系统完整性，该系统无法部署到配备有PatchGuard内核保护机制的64位操作系统上。而随着网络游戏的迅速发展，大部分游戏均在64位系统上进行开发和测试[1]，GameGuard的低兼容性导致其无法为新兴网络游戏提供有效保护。而CheatBlocker核心的内核事件监控功能仅通过Windows原生API实现，无需破坏系统完整性，因而能兼容64位系统，这也是其相较于GameGuard而言最大的优势。EasyAntiCheat虽然可以防御以上所列所有模块注入技术，但是未对用户输入模拟技术进行任何防御。实验表明，CheatBlocker在防御面、防御效果以及兼容性方面均优于已有的应用层及内核层反外挂系统。

### 4.2 性能测试

本文还针对各反外挂系统的性能进行了对比实验。性能测试工具为Windows内置的Performance Monitor，性能指标包括CPU占用、内存占用以及游戏启动时间。为了保证反外挂系统的兼容性，实

**表 3　外挂测试样本**

| | 外挂工具 | 相关外挂技术 | 外挂行为描述 |
|---|---|---|---|
| FIFA 10 | FIFA Cheater 0.5 | CreateRemoteThread 注入 | 内存修改 |
| | Mr.Anti.Fun Cheat | CreateRemoteThread 注入 | 内存修改 |
| | CPY FIFA Cheater | QueueUserApc 注入 | 代码注入 |
| | FIFA Auto Runner | 窗口模拟 | 挂机脚本 |
| CROSS FIRE | Sniper Rifle 1.0 | CreateRemoteThread 注入 | 内存修改 |
| | LOCK Health Cheater | QueueUserApc 注入 | 内存修改 |
| | Ice Modz 6041 Rc1 | Hook Windows 消息注入 | 内存修改 |
| | Crossfire Hacker | 线程劫持注入 | 代码注入 |
| | Remote Dll Injector | 所有注入技术 | DLL注入 |
| | Assassin Wall Cf | 窗口模拟 | 挂机脚本 |
| | Auto-Shooter | 输入法注入/全局模拟 | 挂机脚本 |
| | Antifun GOLD Getter | 线程劫持注入/窗口模拟 | 挂机脚本 |

**表 4　反外挂系统防御效果对比**

| 外挂技术 | 反外挂系统 | | | | | |
|---|---|---|---|---|---|---|
| | CheatBlocker | Nprotect | Xray | Warden | GameGuard | EasyAntiCheat |
| 创建远程线程注入 | √ | √ | √ | √ | √ | √ |
| 插入APC注入 | √ | √ | × | × | √ | √ |
| 线程劫持注入 | √ | √ | × | √ | √ | √ |
| Hook Windows消息注入 | √ | × | √ | × | √ | √ |
| 输入法注入 | √ | × | × | × | √ | √ |
| 全局模拟 | √ | × | × | × | √ | × |
| 窗口模拟 | √ | × | × | × | √ | × |
| 是否支持64位系统 | √ | √ | √ | √ | × | √ |

验在32位系统VM2中进行。由于反外挂系统检测到外挂后会终止游戏进程，因此本文性能测试仅针对不出现外挂的情况。具体实验方法为，首先在不开启任何反外挂系统的情况下，启动游戏并记录游戏启动时间，然后在游戏稳定运行过程中使用Performance Monitor实时监控系统的各项性能指标并记录，最后得出各项性能指标在监控时间段内的平均值。然后开启反外挂系统，同样记录游戏启动时间以及游戏运行稳定后的平均内存占用和平均CPU占用。不开启反外挂系统以及开启每个反外挂系统时的性能采样实验均进行5次，然后将5次实验所得数据取平均值。未开启任何反外挂系统以及开启各个反外挂系统时的平均系统资源占用情况以及平均游戏启动时间如表5所示。可见CheatBlocker的系统开销只略高于用户层反外挂系统，且在内核层反外挂系统中3项指标均为最低，说明CheatBlocker在保证完善的反外挂功能的同时，其引入的性能开销极低，且优于测试中的其他两款内核层反外挂系统。

## 5　结束语

针对当前网络游戏外挂泛滥的情况，本文从操作系统内核层入手，根据目前外挂程序的两种主流类型：进程篡改和用户输入模拟，设计了基于Windows内核事件的反外挂系统CheatBlocker。针对进程篡改类外挂，CheatBlocker使用内核中相关的事件监听函数，通过句柄权限过滤与模块签名校

**表 5　反外挂系统系统开销对比**

| 系统开销 | No Anti-Cheat | CheatBlocker | Nprotect | Xray | Warden | GameGuard | EasyAntiCheat |
|---|---|---|---|---|---|---|---|
| 平均CPU占用 (%) | 23.5 | 28.7 | 25.8 | 26.4 | 23.3 | 30.8 | 29.4 |
| 平均内存占用(%) | 35.3 | 35.8 | 34.7 | 37.5 | 36.5 | 36.7 | 35.8 |
| 平局启动时间(s) | 20.1 | 24.6 | 23.4 | 22.8 | 22.3 | 28.9 | 25.7 |

验来阻止外挂进程对游戏进程进行进程篡改以及代码注入；针对用户输入模拟类外挂，CheatBlocker监测系统中进程的创建事件并使用内核注入技术注入反外挂模块，利用该模块拦截鼠标键盘模拟相关的系统函数，阻止外挂进程向游戏进程发送模拟消息。实验结果证实了CheatBlocker能有效检测并防御上述两类外挂技术，且对比于其他应用层以及内核层反外挂系统，CheatBlocker具有更好的防御效果及兼容性。

## 参 考 文 献


[1] 腾讯游戏研发部游戏安全中心. 游戏安全: 手游安全技术入门[M]. 北京: 电子工业出版社, 2016.
Game Security Center of Tencent Game R & D Department. Game Security: Introduction to Mobile Security Technology[M]. Beijing: Electronic Industry Press, 2016.

[2] YAN J J and CHOI H J. Security issues in online games[J]. *The Electronic Library*, 2002, 20(2): 125–133. doi: 10.1108/02640470210424455.

[3] YAN J and RANDELL B. A systematic classification of cheating in online games[C]. The 4th ACM SIGCOMM Workshop on Network and System Support for Games, New York, USA, 2005: 1–9. doi: 10.1145/1103599.1103606.

[4] KABUS P, TERPSTRA W W, CILIA M, *et al*. Addressing cheating in distributed MMOGs[C]. The 4th ACM SIGCOMM Workshop on Network and System Support for Games, New York, USA, 2005: 1–6. doi: 10.1145/1103599.1103607.

[5] CHOI Y, CHANG S J, KIM Y, *et al*. Detecting and monitoring game bots based on large-scale user-behavior log data analysis in multiplayer online games[J]. *The Journal of Supercomputing*, 2016, 72(9): 3572–3587. doi: 10.1007/s11227-015-1545-2.

[6] 罗平, 徐倩华. 网络游戏外挂技术及检测[J]. 计算机工程与设计, 2007, 28(6): 1273–1276. doi: 10.3969/j.issn.1000-7024.2007.06.011.
LUO Ping and XU Qianhua. Hack technology and detection of online games[J]. *Computer Engineering and Design*, 2007, 28(6): 1273–1276. doi: 10.3969/j.issn.1000-7024.2007.06.011.

[7] 杨英杰, 冷强, 常德显, 等. 基于属性攻击图的网络动态威胁分析技术研究[J]. 电子与信息学报, 2019, 41(8): 1838–1846. doi: 10.11999/JEIT181025.
YANG Yingjie, LENG Qiang, CHANG Dexian, *et al*. Research on network dynamic threat analysis technology based on attribute attack graph[J]. *Journal of Electronics & Information Technology*, 2019, 41(8): 1838–1846. doi: 10.11999/JEIT181025.

[8] CHANG H and ATALLAH M J. Protecting software code by guards[C]. ACM CCS-8 Workshop DRM on Security and Privacy in Digital Rights Management, Berlin, Germany, 2001: 160–175. doi: 10.1007/3-540-47870-1_10.

[9] THE L B and KHANH V N. GameGuard: A windows-based software architecture for protecting online games against hackers[C]. The Symposium on Information and Communication Technology, Hanoi, Vietnam, 2010: 171–178. doi: 10.1145/1852611.1852643.

[10] 梁光辉, 庞建民, 单征. 基于代码进化的恶意代码沙箱规避检测技术研究[J]. 电子与信息学报, 2019, 41(2): 341–347. doi: 10.11999/JEIT180257.
LIANG Guanghui, PANG Jianmin, and SHAN Zheng. Malware sandbox evasion detection based on code evolution[J]. *Journal of Electronics & Information Technology*, 2019, 41(2): 341–347. doi: 10.11999/JEIT180257.

[11] WOO J, KANG A R, and KIM H K. The contagion of malicious behaviors in online games[J]. *ACM SIGCOMM Computer Communication Review*, 2013, 43(4): 543–544. doi: 10.1145/2534169.2491712.

[12] AHMAD M A, KEEGAN B, SRIVASTAVA J, *et al*. Mining for gold farmers: Automatic detection of deviant players in mmogs[C]. 2009 International Conference on Computational Science and Engineering, Vancouver, Canada, 2009: 340–345. doi: 10.1109/cse.2009.307.

[13] KWON H, MOHAISEN A, WOO J, *et al*. Crime scene reconstruction: Online gold farming network analysis[J]. *IEEE Transactions on Information Forensics and Security*, 2017, 12(3): 544–556. doi: 10.1109/tifs.2016.2623586.

[14] CHUNG Y, PARK C Y, KIM N R, *et al*. Game bot detection approach based on behavior analysis and consideration of various play styles[J]. *ETRI Journal*, 2013, 35(6): 1058–1067. doi: 10.4218/etrij.13.2013.0049.

[15] DUH H B L and CHEN V H. Cheating behaviors in online gaming[C]. The 3rd International Conference on Online Communities and Social Computing, Berlin, Germany, 2009: 567–573. doi: 10.1007/978-3-642-02774-1_61.

[16] 傅建明, 彭碧琛, 杜浩. 一种组件加载漏洞的动态检测[J]. 清华大学学报: 自然科学版, 2012, 52(10): 1356–1363, 1369. doi: 10.16511/j.cnki.qhdxxb.2012.10.007.
FU Jianming, PENG Bichen, and DU Hao. Dynamic detection of component loading vulnerability[J]. *Journal of Tsinghua University: Science and Technology*, 2012, 52(10): 1356–1363, 1369. doi: 10.16511/j.cnki.qhdxxb.2012.10.007.

[17] HOGLUND G and MCGRAW G. Exploiting Online Games: Cheating Massively Distributed Systems[M]. New York, USA: Addison-Wesley Professional, 2007: 119–125.

[18] WEBB S D and SOH S. Cheating in networked computer

games: A review[C]. The 2nd International Conference on Digital Interactive Media in Entertainment and Arts, Perth, Australia, 2007: 105–112. doi: 10.1145/1306813.1306839.

[19] LIU H I and LO Y T. DaCAP-a distributed Anti-Cheating peer to peer architecture for massive multiplayer on-line role playing game[C]. The 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID), Lyon, France, 2008: 584–589. doi: 10.1109/ccgrid.2008.49.

[20] SEBASTIO S, AMORETTI M, MURGA J R, *et al*. Honest vs Cheating Bots in PATROL-based Real-time Strategy MMOGs[M]. CAGNONI S, MIROLLI M, and VILLANI M. Evolution, Complexity and Artificial Life. Heidelberg: Germaay, Springer, 2014: 225–238. doi: 10.1007/978-3-642-37577-4_15.

傅建明：男，1969年生，教授，研究方向为恶意代码检测和漏洞检测与防御.

杨　铮：男，1995年生，硕士生，研究方向为系统安全.

罗陈可：男，1996年生，硕士生，研究方向为系统安全与二进制安全.

黄坚伟：男，1996年生，硕士生，研究方向为网络安全.

责任编辑：马秀强