计算机辅助测验中基于消息监控与包过滤技术的监考方法

张新林
（罗定职业技术学院　罗定 527200）

摘　要　在机考试时，为防止考生利用计算机作弊，针对考生启动非法程序获取资料的情况，使用钩子函数对系统和进程监控，捕捉到非法程序在创建、激活之时即予以强制关闭。本文针对利用网络进行信息传递的情况，使用包过滤技术，把不属于本机与考试服务器间通信的非法数据包进行过滤。对所有作弊行为警告，收集作弊证据并传送到服务器，实现了人工监考所不能达到的目的。还给出了具体实现的关键代码。

关键词　计算机辅助测验，监控，钩子函数，包过滤

Invigilation Plan Based on the Message Surveillance and Package Filtration Technique

ZHANG Xin-Lin
(Luoding Polytechnic, Luoding 527200)

Abstract　Based of computer cheating methods and aiming to the acquirement of the information by illegal program, the lawfulless program can be captured and closed forcibly when it is created and activated with the surveillance of system and process by hook function. In virtue of network information transferring , the illegal data package which doesn't belong to the exchange between the local computer and the exam server are supposed to be cancelled by package filtration technique. The invigilation plan will warn all the cheating examees, collect the cheating evidences, transfer them to the server and achieve the goal that mannul invigilation can't have. As the same time, the key specific codes are given in the text.

Keywords　CAT(Computer-Assisted Testing), Surveillance, Hook function, Package filtration

随着计算机技术的发展、快速以太网的广泛应用，教育评价已转移到计算机上进行，实行无纸化考试。计算机辅助测验(CAT：Computer-Assisted Testing)已成为计算机教育和教育技术领域内的一个研究热点，它从评价内容、评价方法和评价形式等多方面发展了传统测评理论和实践。现在各种计算机等级考试、高等学校英语应用能力 AB 级、信息技术教育考试等已实行在机考试的实时评价计分方式。

在传统的考试中，监考员只需监视考生的举动即可。但在机考试与传统考试的监考相比就面临较多的问题。首先，在机房考试，由于有机器的阻隔，对监考员监视考生的行为有一定的影响；其次，考生在自己的机器上操作，至于操作什么，监考员就较难观察到了。当监考人员是非计算机专业人员，较缺乏计算机知识时，就更难判别考生是在干什么了，此时考生就很有可能利用计算机进行作弊了。

因此，利用计算机自身进行考试监控，对考试过程的实时控制并记录操作行为就显得非常重要了，在考试系统中加入实时监控功能是亟待解决的问题。目前还没有专门论及计算机监考方法的文献，本文着重研究利用计算机监考的实施方案。

在考试时，考生利用计算机技术进行作弊的方法主要有两种情况：启动在考试时禁用的软件，如在考英语时，私自调用翻译软件、启动工具软件；利用网络传递信息。

对于运行非法软件的情况，我们只要监视系统及其他应用程序即可，除考试允许的进程之外的其他进程一律封杀。利用网络进行传输信息，就要监视本机收发信息的源头与目的地址。若是本机与考试服务器进行通讯，则允许，否则一律禁止。

1　对消息监控，禁止运行非法软件

Windows 操作系统是建立在事件驱动的机制上的，也就是整个系统都是通过消息的传递来实现的[1]。钩子(hook)是 Windows 系统中非常重要的系统接口，用它可以截获并处理送给其他进程的消息。钩子可以监视 Windows 自身操作系统和进程中的各种事件消息，截获发往目标窗口的消息并进行处理。这样，我们就可以在系统中安装自定义的钩子，监视系统中特定事件的发生，完成特定的功能，比如截获键盘、鼠标的输入，其他进程的运行情况等。由它可以实现对考试过程的全程监控，达到利用计算机自身进行监考的目的。

1.1　利用钩子函数实现对消息监控，禁止违规进程运行

钩子函数的种类很多，要监视系统进程，选用类型为 WH_SHELL 的钩子较为适合，因为当外壳应用程序是激活的，并且当顶层窗口建立或者销毁时，系统都会调用 WH_SHELL Hook 子程。也就是说，WH_SHELL 钩子可监视所有应用程序的主窗口创建或者关闭[1]。在考试过程中，用 WH_SHELL 钩子始终监视系统的所有进程。若发现有非法程序运行，则记录其违规行为，并强制关闭该程序，达到监视目的。

1.2　利用钩子函数监视系统的要点

由于要监视系统的进程，WH_SHELL 钩子必然是全局的，必须把这个钩子定义到 DLL 的动态链接库里，即要创建

张新林　副教授，主要研究方向：网络、网络数据库、应用软件。

一个动态链接库的文件。

由于钩子是要安装到系统中，并运行在操作系统里面，因此这个钩子不能使用考试系统(或监控软件)所定义的任何变量。利用共享内存技术，就能巧妙地解决此问题，用 API 函数 CreateFileMapping() 创建一个共享内存，任何运行的程序都可以使用这块内存[2]。

使用白名单文件区分哪些进程是可以允许考生运行的合法程序。凡不在白名单文件列表中的进程一旦运行，即为非法。

若监控功能放在考试系统中，则考生是不会主动杀死考试系统的，杀死则意味着不能考试了。但若监控功能不在考试系统的内部，而是一个独立的监控软件时，就要防止被杀死。最好的办法就是隐形，在任务栏和任务管理器里面均找不到监控进程，这样考生就无法杀死监控进程了，或者根本不知道在监视他(她)。只要在主程序 Application->Run()的前面加上 SetWindowLong(Application->Handle, GWL_EXSTYLE, GetWindowLong(Application->Handle, GWL_EXSTYLE)|WS_EX_TOOLWINDOW)就能达到隐形目的了。

1.3 实现系统监视的主要代码

动态链接库部分：先采用“PUBLIC_MEMORY”定义共享内存标志，TSharedMemory AppMem(APPMARK,4096)；产生在监控系统里共享的内存，HookSharedData->hHook = hThisHook，把 hThisHook 保存到共享内存里。同时要定义共享的数据结构，保存钩子的句柄 HHOOK hThisHook 等。

若是独立的监控，注意使用 GetWindowLong(Application->Handle, GWL_EXSTYLE)|WS_EX_TOOLWINDOW)隐身，防止被考生察觉而杀死。若监控功能直接放在考试系统中，则不必隐身。

主要监控过程如下：

```
#define APPMARK "PUBLIC_MEMORY"//先实行动态链接部分，共享内存标志
……
typedef struct //定义共享的数据结构
{HHOOK hHook; //当前使用的 HOOK
} THookSharedData;
……
hThisHook = SetWindowsHookEx(WH_SHELL, (HOOKPROC) HookedShellProc, HInstance, 0);
TSharedMemory AppMem(APPMARK,4096); //在监控系统里共享的内存
THookSharedData * HookSharedData
((THookSharedData * )(AppMem. AppInfo->Data));
HookSharedData->hHook = hThisHook; //把 hThisHook 保存到共享内存里}
……
    if(AppMem. Valid)if(AppMem. Exists) //如果共享内存存在
     { HWND hMainWnd = AppMem. AppInfo->hMainForm;
    if(hMainWnd){
    if(nCode==HSHELL_WINDOWCREATED) {
    PostMessage(hMainWnd, WM_USERCMD, UC_WINHOOK, wParam); }}}
  //在 Hook 里无法调用 hThisHook，必须用共享内存里面的 hHook
THookSharedData * HookSharedData
((THookSharedData * )(AppMem. AppInfo->Data));
return CallNextHookEx ( HookSharedData-> hHook, nCode, wParam, lParam);}
主程序部分：
……
SetWindowLong(Application->Handle, GWL_EXSTYLE, GetWindowLong(Application->Handle, GWL_EXSTYLE)|WS_EX_TOOLWINDOW); //隐身，若是独立的监控才需要，监控功能直接放在考试系统中则无必要。
Application->Run();
    ……
TSharedMemory AppMem(APPMARK, 4096); //定义共享的内存
MyHook = new THookedProcs;
MyHook->InitFuncs()//安装钩子。
if(! ValidAppCheck(hWnd, szWinClass, szWinCaption)) //调用 ValidAppCheck
s= DateTime +“关闭禁用程序”;//收集作弊证据并发送给服务器的违规资料。
Bool_fastcall TFormMain::ValidAppCheck(HWND hwnd, char * cls, char * cap)
{ TBinFile f;
    f. FileName = “白名单文件”; {……
    while(fgets(aLine,2000,f)) //读白名单，下面用类名及标题名分别判断是否允许该进程运行。
    {if(strnicmp(aLine,“CLASS=”,6)==0) //用类名识别{
    if(stricmp(aLine+6,cls)()0) {
    PostMessage(hwnd, WM_CLOSE, 0, 0); //关闭程序
    return false; } }
    else if(strnicmp(aLine,“CAPTION=”,8)==0) //用标题识别
    {if(stricmp(aLine+8,cap)()0) {
    PostMessage(hwnd, WM_CLOSE, 0, 0); //关闭程序
……
```

2 监视网络，利用包过滤禁止非法信息的传递

在大多数情况下，例如在真实环境下的考试、B/S 结构的考试系统中，是允许考生打开资源管理器的，即上面的消息监控功能认为访问网上邻居是正常的，存在利用网络作弊的可能性。因此，要对网络进行独立监视，只需监视本机的网络通信情况，无需关心网络上的其他机器。凡是本机与服务器间的数据包通信，不管是采用何端口，均视为合法，但本机与非服务器的通信一律视为非法。这样，只要提取本机的数据包源地址、目的地址，抛弃含有非法地址的数据，就能达到阻止利用网络进行作弊的行为了。

2.1 数据包的获取与结构分析

为了容易实现及通用性，假定通信协议使用 TCP/IP 协议。首先创建原始套接字，通过 setsockopt()设置 IP 头操作选项，利用 bind()函数将原始套接字绑定到本地网卡。此时即可对本机的数据包进行嗅探，通过 recv()函数完成对原始数据包的提取。这些原始套接字包含有 IP 头、TCP 头等信息。我们的目的就是要提取数据包的源地址与目的地址，只要是非法的就抛弃。而对于其他的信息，就无需关心。IP 数据报(IPv4)的结构如图 1[3]。

于是定义一个数据结构来表示此 IP 数据段头，具体如下。

```
typedef struct _IP{
union{ BYTE Version; //版本
BYTE HdrLen;// 头长度};
BYTE ServiceType; // 服务类型
WORD TotalLen; // 总长
WORD ID; // 标识
union{ WORD Flags; // 标志
WORD FragOff; // 分段偏移};
BYTE TimeToLive; // 生命期
BYTE Protocol; // 协议
WORD HdrChksum; // 头校验和
DWORD SrcAddr; // 源 IP 地址
DWORD DstAddr; // 目的 IP 地址
BYTE Options; // 选项} IP;
typedef IP * LPIP;
typedef IP UNALIGNED * ULPIP;
```

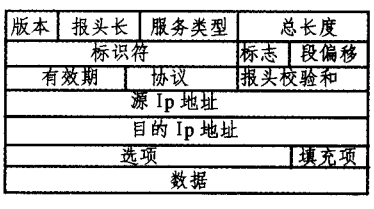

图 1

2.2 主要代码

sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)); // 创建原始套接字

setsockopt(sock, IPPROTO_IP, IP_HDRINCL, (char *)&flag, sizeof(flag)); // 设置 IP 头操作选项，其中 flag 为 true，亲自对 IP 头进行处理

pHost = gethostbyname((char *)LocalName)); // 获取本地 IP 地址

addr_in.sin_addr = *(in_addr *)pHost->h_addr_list[0]; //IP

addr_in.sin_family = AF_INET;

addr_in.sin_port = htons(57274);

bind(sock, (PSOCKADDR)&addr_in, sizeof(addr_in)); // 把原始套接字 sock 绑定到本地网卡地址上[4,5]。

此时可通过 recv()函数从网卡接收数据。接收到的原始数据包存放在缓存 RecvBuf[]中,缓冲区长度 BUFFER_SIZE 定义为 65535。根据前面对 IP 数据包进行分析,获取源地址与目标地址:

int ret = recv(sock, RecvBuf, BUFFER_SIZE, 0); // 接收原始数据包信息

if (ret > 0) {

ip = *(IP*)RecvBuf; // 对数据包进行分析,并输出分析结果

if (inet_ntoa(*(in_addr*)&ip.SrcAddr))<>addr_in.sin_addr)

or (inet_ntoa(*(in_addr*)&ip.DstAddr))<>addr_in.sin_addr)

or (inet_ntoa(*(in_addr*)&ip.SrcAddr))<>server.sin_addr)

or (inet_ntoa(*(in_addr*)&ip.DstAddr))<>server.sin_addr) //判断源地址、目标地址是否为本机或服务器的 IP,这里若为了程序的灵活性,可采用 IP 地址的黑白名单的办法。

{close(sock);//禁止通信。}}

结束语 计算机辅助测验(CAT)具有广泛的应用前景,在考核过程中利用计算机自身的监控能力来进行监考是人力所无法达到的。在实际应用中,要注意及时卸载钩子,以免占用资源。采集数据时不要把网卡设置为混杂模式,否则收到网络上的其他数据,达不到监视目的,同时使网速变慢。凡有违规行为,均要记录并及时警告考生,达到某种程度,应中止该考生的考试。以上方法在 100M 局域网内(61 台计算机),Windows2000 下测试通过。

参 考 文 献

1 Richter J(美). Windows 核心编程[M]. 王建华,译. 北京:机械工业出版社,2000

2 赵新宇,林作铨. 具有监控能力的 Agent 模型[J]. 计算机科学,2006(3):11~17

3 陈少辉,张艳宁,刘艳玲. 基于封包截获技术的个人防火墙核心驱动技术[J]. 计算机工程,2007(6):123~125

4 胡金初. 数据包时延及控制策略的研究[J]. 计算机科学,2006(11):52~53

5 刘翔,席守卿,吴昕怡,等. Windows2000 系统中网络数据包截获方法[J]. 武汉理工大学学报,2002(24):88~89

(上接第 276 页)

表 3 权限及其 DAE

Permission	DAE
P1:design test case and data	Requirements: professional in JAVA, two years' test experience, professional in ORACLE, familiar with test theory, not a member of the team of module B. language=JAVA AND testing_experience ≥2 AND database=ORACLE AND familiar_with_test_theory=yes AND current_program _module≠B
P2:config test environment	Requirements: familiar with at least one tool, professional in JAVA and ORACLE, not a member of the team of module B. familiar_test_tool≥1 AND language=JAVA AND database=ORACLEANDcurrent_program _module≠B
P3:perform test	Requirements: familiar with at least one tool, professional in JAVA and ORACLE, familiar with test theory, two years' test experice, not a member of the team of module B. familiar_test_tool≥1 AND testing_experience≥2 AND language=JAVA AND database=ORACLE AND familiar_with_test_theory=yesANDcurrent_program _module≠B

参 考 文 献

1 Sandhu R, Coyne E, Feinstein H, et al. Role-based Access Control Models. IEEE Computer, 29(2): 38~47

2 Barka E, Sandhu R. Framework for Role-based Delegation Models. In: Proc. of 16th Annual Computer Security Application Conference (ACSAC2000). New Orleans, USA: IEEE Computer Society Press, 2000

3 Barka E, Sandhu R. A role-based delegation model and some extensions. In: Proc. of 23rd National Information Systems Security Conference (NISSC 2000). Baltimore, USA, 2000

4 Zhang Longhua, Ahn Gail-Joon, Chu Bei-Tseng. A rule-based framework for role-based delegation. In: Proc. of SACMAT'01. Chantilly, VA, USA: ACM Press, 2001

5 Tamassia R, Yao Danfeng, Winsborough W H. Role-based cascaded delegation. In: Proc. of the SACMAT'04. Yorktown Heights, New York, USA: ACM Press, 2004

6 Zhang Xinwen, Oh Sejong, Sandhu R. PBDM: A Flexible Delegation Model in RBAC. In: Proc. of the SACMAT'03. Como, Italy: ACM Press, 2003

7 赵青松,孙玉芳,孙波. RPRDM:基于重复和部分角色的转授权模型. 计算机研究与发展,2003,40(2):221~227

8 Ye Chun-xiao, Fu Yun-qing, Wu Zhong-fu. An Attribute-based Delegation Model and Its Extension. Journal of Research and Practice in Information Technology, 2006,38(1):3~17

9 叶春晓,吴中福,符云清,等. 基于属性的扩展委托模型. 计算机研究与发展, 2006,43(6):1050~1057

10 叶春晓. 基于角色访问控制(RBAC)中属性约束委托模型研究:[博士论文]. 重庆:重庆大学,2005