1575与6.4病毒杂合的新病毒
——“Bloody”病毒的发现和清除

罗　辉　（湖南省双峰工商银行）

编者按:有人利用计算机进行反动宣传,必须予以揭穿。罗辉同志的文章对“bloody”病毒进行了彻底披露,并给出清除方法,值得同行借鉴。

最近一段时间,连续有几个单位的机器感染了同一种病毒来求诊。该病毒源码长度为7C5H(1989)字节,由于病毒发作时显示"Bloody....."之类的信息,笔者姑且称之为Bloody病毒。因该病毒有一定的欺骗性,目前所有的清毒软件对它都出现误诊,本文特对它作一番分析,并给出相应的清除方法。

一、Bloody病毒的特点

使用市面上流行的许多新版清毒软件扫描系统,有些对感染Bloody病毒的文件还是有反应的,但是都不能正确清除该病毒。譬如,用CPAV 2.0扫描将报告发现"F Virus",但不能清除之;用KILL70软件扫描之,则报告发现Paralyz eye病毒,但选用CLEAN功能清除之,宿主文件长度却截成0,或不能运行;而用SCAN100软件扫描之,报告发现1575/1591病毒,清除后,宿主程序不能运行;等.....几乎所有已有的清毒软件都将这种病毒报告为1575病毒或其变种。然而,该病毒发作时,却表现出酷似90年流行的6.4病毒的表现特性:显示一些反动的宣传性文字,显示信息都差不多,如"Bloody! June 4th"之类。通过同时分析1575病毒和6.4病毒的源码,可以发现,Bloody病毒实质上是将1575病毒和6.4病毒合二为一而成的一种新的变种病毒,借用1575病毒的执行代码,稍作修改和变通,同时显示6.4病毒中的反动文字。其病毒体源码与1575病毒十分相似。

1.7C5H病毒与1575病毒一样,都对COM和EXE可执行文件进行感染。在感染COM文件时,都对COM文件的开始部分修改,以实现在该宿主程序执行之初即由病毒夺得控制权。对COM型染毒文件,开始代码都是:

```
PUSH CS
MOV AX,CS        ;修改执行代码的段址CS,指向病毒体所在段,其中的星号
ADD AX,* * * *   ;表示为原正常宿主文件长度以节为单位调整后的文件长度
PUSH AX          ;将病毒体段址压入堆栈
MOV AX,0100
PUSH AX          ;将病毒体所在段偏移IP压入堆栈
RETF             ;通过远程返回指令弹出CS:IP,即使程序转到病毒代码继续
```

由于目前许多清毒软件都是用这段代码作为1575/1591/Paralyz eye病毒的特征代码加以识别,因此这些清毒软件扫描到Bloody病毒时,都错误地报告为1575/1591/Paralyz eye病毒。

同时,由于Bloody病毒与1575病毒对原COM型宿主文件的开始0CH(对EXE型文件是18H)字节代码的保存位置不同,Bloody病毒将它保存在病毒体偏移0131H开始的0CH字节单元内,而1575病毒则保存在病毒体偏移010BH开始的0CH字节单元内;而且对EXE文件的感染,对原宿主文件的文件头程序入口处的代码段段址CS、段偏移IP、堆栈段段址SS、堆栈段偏移SP及原文件的大小等信息,Bloody病毒是保存在病毒体偏移的0149H-14AH(原CS)、14BH-14CH(原IP)、157H-158H(原SS)、159H-15AH(原SP)及14DH-150H(原文件的长度)等单元,而1575病毒则将它们保存在123H-124H(原CS)、125H-126H(原IP)、131H

-132H (原 SS)、133H- 134H (原 SP)及 127H- 12AH (及原文件长度)等单元,因而有些清毒软件在将 Bloody 病毒当做 1575 病毒清除时,由于有关原文件的重要信息的获取位置出错,导致清毒失败,原文件遭到破坏。

2.Bloody 病毒类似于 1575 病毒,也首先特别感染 C:\COMMAND.COM 文件,并且都将"C:\COMMAND.COM"字串加密存放,不过是 Bloody 病毒将该字串的 ASCII 码减 10H 存放在病毒码偏移 1B3H 处,而 1575 病毒将该字串减 20H 存放在病毒码偏移 20AH 处。

3.Bloody 病毒也不通过修改内存大小进驻内存高端,而是通过修改内存控制链动态进驻内存的。当已被染毒的 COM 或 EXE 型宿主程序执行开始,都由病毒控制首先将指向程序段前缀 PSP 的起始段地址的 ES 保存起来,然后将 ES 的值减一,即使 ES 指向内存块的控制块段地址,再判断该块是否是最后一块内存块。只有当本内存块是最后一块内存块时,才让病毒驻留内存;否则放弃,不驻留内存。若该内存块是最后一块,则将病毒体占用长度 8D5H(病毒代码长度 7C5H 加 110H 的堆栈和数据运算空间)右移四位,得到病毒体将占用的内存节数(1 节为 16Byte),与本内存块可分配的内存节数比较,如果可分配的内存节数小于 8DH,即不足以容纳病毒体,则不驻留内存;否则,则修改本次内存块可分配节数(减少 8DH),并将最高内存块下移 8DH,从而在本内存块高端挤出 8DH 节的空间,将病毒体驻留在其中。这样,该病毒占用了本宿主程序的可用内存空间,但同时也使病毒内存消耗量降低到了最少。

4.Bloody 病毒接管 INT21H 中断服务,它控制了 INT21H 的 11H、12H、1AH、57H 功能号,其中 57H 功能用于本 INT21H 内部功能的切换使用。通过控制 1AH 功能得到磁盘传送区地址,控制 11H 和 12H 功能得到将被感染的可执行文件名,并进行感染。该病毒将对 A-D 逻辑盘上的可执行文件进行感染,对 E 盘及以上逻辑盘上的文件将不予感染。从上可以看出,该病毒传染速度很快,在你使用诸如 DIR 命令之类的调用 INT21H 的 11H 和 12H 的命令时,将会迅速大批量的对可执行文件进行感染。

5.Bloody 病毒也是通过控制 INT1CH 中断和一些控制单元决定是否和如何发作的。Bloody 病毒使用病毒体偏移 161H 单元作为病毒发作的触发器。每在传染了一个新系统时,将使新系统中病毒体内该触发器的值减一。在修改后的 INT21H 中断服务子程序中,将判断该触发器的内容是否是 64H,是则发作。病毒发作时并不对磁盘和内存资源进行破坏,但将显示大段带有政治目的的反动文字,与以前的 BOOT 型 6.4 病毒的显示信息及其相似。

6.Bloody 病毒和 1575 病毒一样,都是利用染毒文件最后两字节作为染毒标志,但前者的标志位内容是 1989H,而后者的标志位内容是 0A0CH。

二、Bloody 病毒的检测

1.内存的检测

用 DEBUG 的 D 命令查 0:84H-0:87H,如果内存已感染了 Bloody 病毒,则 0:84H-0:85H 单元内容应是:0439H,这是修改后的 INT21H 中断服务子程序的入口偏移;由于 INT21H 的中断服务子程序与病毒体所在段址是一样的,你可将 0:86H-0:87H 两单元的内容(****)作为段址(记住:这两单元是低位在前,高位在后,作为地址使用时,需高低位互换)。然后使用命令:D ****:08C3,08C4 查看 08C3H-08C4H 的内容,如果这两个单元的内容为 1989H,则几乎可以肯定内存中已经感染了 Bloody 病毒。

2.文件的检测

对 COM 和 EXE 文件,可用 DEBUG 察看该文件(如是 EXE 文件需先将文件名更名为非 EXE 文件名)最后两字节内容是否是 1989H,同时查看文件后部是否有"Bloody! June 4th, Made in Chengdun the eyes, you're paralyz..."之类的信息,如有,则肯定感染了 Bloody 病毒。

三、Bloody 病毒的清除

1.内存病毒的清除

用 DEBUG 仿内存病毒的检测方法找到病毒体的段址。将该段址所在段内偏移 0129H-012CH 中保存的正常 INT21H 中断向量的内容写入 0:84H-87H 处,将段内偏移 012DH-0130H 中保存的正常 INT1CH 中断向量的内容写入 0:70H-73H 处,即可清除。

2.对 COM 文件的清毒步骤

C>DEBUG <染毒文件.COM>
-D　　;记下当前所在段址,记为 CS1
-R CX
CX * * * *　　;将 CX 中的文件长度减去 7C5H,得到原文件长度
:　　;将原文件长度写入冒号后面,即恢复原文件长
-G=100,10B　;进入病毒体所在段.将新的段址记为 CS2
-R DS　　;修改数据段段址指向 CS2
DS * * * *
:<CS2>
-M 131L0C,CS1:100 ;将保存在病毒体偏移 131H 处开始的原正常
　　;文件开始的 0CH 字节内容回送到原文件开头.
-W　　;写文件
-Q

3.对 EXE 文件的清毒步骤

C>DEBUG <染毒文件.EXE>
-D CS:0149 L4　;149H-14AH 为原 CS 内容,14BH-14CH 为原 IP 内容.
-D CS:0157 L4　;157H-158H 为原 SS 内容,158H-159H 为原 SP 内容
-D CS:014D L4　;14DH-14EH 为原文件长度高位,
　　;14FH-150H 为原文件长度低位
-Q

将文件长度除以 200H(即 512B),所得商+1 即为原文件所占扇区数,记为 SECE;所得余数为最后一扇区的实际字节数,记为 BYTE.

C>REN <染毒文件.EXE> <染毒文件>
C>DEBUG <染毒文件>

用 E 命令将 SECE 送 CS:104H-145H,将 BYTE 送 CS:102H-103H,将原 CS 送 CS:116H,原 IP 送 CS:114H,原 SS 送 CS:10EH,原 SP 送 CS:110H。

-W　　;写文件
-Q
C>REN <染毒文件> <染毒文件.EXE>

有必要指出的一点是:染毒文件清毒后,一般很难完全恢复原文件的长度,而是与原文件长度有 0H-0FH 字节(即一节)的出入。这是任何清毒软件也不可避免的遗憾,事实上许多清毒软件在感染 1575 病毒或其它类似病毒的文件清毒过程中早已出现了这一现象。但这完全不影响原文件的正常执行。造成这一现象的原因是:Bloody 病毒在感染 COM 和 EXE 可执行文件时,首先都将原文件长度调整(文件尾不足一节即 10H 字节的,添足为 10H 字节)为以节为单位的倍数,以便附加的病毒体代码以新的一节开始存放。对 EXE 型染毒文件,保存在病毒体偏移 014DH-0150H 四个单元处的文件长度即是已被调整后的文件长度。由于原文件完全精确的长度已无法再获得,因此我们只有根据调整以后的文件长度对宿主文件进行修复。故出现清毒后的文件长度与原文件长度有 0-0FH 字节出入的现象。

为方便起见,笔者用 TURBO C 2.0 编制了一个自动清除 Bloody 病毒的 C 程序,将之编译并连接成一个 EXE 文件,即可随时使用。

C>TYPE KBLOODY.C<CR>

```
#include <conio.h>
#include <dos.h>
#include <string.h>
#include <stdio.h>
#include <dir.h>
#include <process.h>
#include <io.h>
#include <fcntl.h>
#include <ctype.h>
int checkmemo();
void xremove(char * path,struct ffblk * fblk);
int scan_kill(int com,char * filename,struct ffblk * flk);
struct ffblk * ffblk;

void main(int argc,char * argv[]) {
  char pathname[80];
  char cur_path[80];
  printf("\n\007\t\t\tThe Killer of Bloody Virus\n");
  printf("\t\t==========\n");
  printf("\t\tCopyright 1994.5.20 Luohui\n\n");
  printf("Usage:\n"); printf("------\n");
  printf("\t%s　<drive:>[\\path　[\\filename.　ext]]\n\n\n",argv[0]);
  printf("\nEnter any key contiune ....."); getch();
  if( argc>1) strcpy(pathname, argv[1]);
  else {
    printf("\nPlease Input Scan Virus Directory:");
    scanf("%s",pathname);
  }
  strupr(pathname);
  if (checkmemo()) {
    sound(200);
    printf("\n Found Bloody virus active in memory!\n");
    printf("Please reboot the system with a clean disk!\n");
    exit(1);
```

```
    }
    if ((pathname[1]= =':') &&(strlen(pathname)= =2)) {
      getcurdir(pathname[0]-64,cur_path);
      strcat(pathname,"\\"); strcat(pathname,cur_path);
      ffblk->ff_attrib=0x10;
    }
    if ((pathname[1]= =':') && (pathname[2]= ='\\')
 (strlen(pathname)= =3)) {
      pathname[2]='\0'; ffblk->ff_attrib=0x10;
    }
    else {
      if (findfirst(pathname,ffblk,0x3f)) {
        printf("\n\007Error: path not found!\n");
        exit(1);
      }
    }
    printf("\n%s\n",pathname); delay(1000);
    xremove(pathname,ffblk);
}
void xremove(char * path,struct ffblk * fblk){
    char newpath[80];
    struct ffblk blk;
    int exe=0,com,retry,print_predir,find_result;
    if ((fblk->ff_attrib0x10)= =0x10) { /* directory */
      strcpy(newpath,path);
      strcat(newpath,"\\*.*");
      if (findfirst(newpath,&blk,0x3f)= =0){
                                    i                f
((strcmp(blk.ff_name,".")!=0)&&(blk.ff_attrib!=FA_LABEL)) {
          strcpy(newpath,path);
          strcat(newpath,"\\");
          strcat(newpath,blk.ff_name);
          xremove(newpath,&blk);
        }
      }
    }
    else {
      if (strstr(fblk->ff_name,".EXE")){
        exe=1; com=0;
      }
      else if (strstr(fblk->ff_name,".COM")){
        exe=1; com=1;
      }
      if (exe){ /* Execute file */
        printf("\nScanning %s....",fblk->ff_name);
        switch (scan_kill(com,path,fblk)) {
          case 1: printf("Found Bloody virus! virus cleaned!\n");
                  sound(200); delay(800); break;
          case-2:printf("Open %s file failure!\n",fblk->
ff_name);
                  sound(300); delay(500); break;
          case-1:printf("Found Bloody virus!cann't cleaned!
\n");
                  sound(400); delay(1000); break;
          case 0: break; /* no virus */
        } nosound();
      } return;
    }
    find_result=findnext(blk);
    print_predir=0;
    while (find_result= =0){
      if((strcmp(blk.ff_name,"..")!=0) & & (blk.ff_attrib!
=FA_LABEL)){
        if (print_predir) printf("\n%s\n",path);
        strcpy(newpath,path);
        strcat(newpath,"\\");
        strcat(newpath,blk.ff_name);
      if ((blk.ff_attrib0x10)= =0x10) {
          print_predir=1;
          printf("\n%s\n",newpath);
        }
        else print_predir=0;
        xremove(newpath,&blk);
      }
      find_result=findnext(&blk);
      if((blk.ff_attrib&0x10)= =0x10)print_predir=0;
    }
    }
    int scan_kill(int com,char * filename,struct ffblk * flk) {
    int handle,set_fattr=0,removed=0;
    struct ftime filetime;
    unsigned char p[0x0c],p1[0x18];
    unsigned int i,j;
    unsigned int flag;
    unsigned long size;
    size=flk->ff_fsize;
    if ((flk->ff_attrib&0x01)= =0x01){
       chmod(filename,1,flk->ff_attrib&0xfe);
      set_fattr=1;
    }
    if((handle=open(filename,O_RDWR O_BINARY))
= =-1){
      if(set_fattr) chmod(filename,1,flk->ff_attrib);
      removed=-2; goto end;
    }
```

```
  getftime(handle,&filetime);
  lseek(handle,size-2,0);
  if (read(handle,&flag,2)!=2) {
    removed=-2; goto end;
  }
  lseek(handle,size-0x2E9,0); / * move to cs:5dcH * /
  if (read(handle,p,0x0C)!=0x0C) { / * read info 'Bloody' * /
    removed=-2; goto end;
  }
  if((flag==0x1989)&&(p[0]=='B')&&(p[2]=='l')&&(p[4]=
='o')
      &&(p[6]=='o')&&(p[8]=='d')&&(p[0x0a]=='y')) {
    removed=-1;
    if (com){ / * Com files * /
      lseek(handle,size-0x794,0); / * move to cs:131H * /
      if (read(handle,p,0x0c)!=0x0c) goto end;
      lseek(handle,0,0);
      if (write(handle,p,0x0c)!=0x0c) goto end;
      lseek(handle,size-0x7c5,0);
    }
    else { / * EXE files * /
      lseek(handle,0,0);
      if (read(handle,p1,0x18)!=0x18) goto end;
      lseek(handle,size-0x77C,0); / * move to cs:149H unit
* /
      if((read(handle,&(p1[0x16]),2)!=2)||(read(handle,&(p1[0x1
4]),2)!=2))
                              / * read old cs & old ip * /
        goto end;
      i=(size-0x7c5) % 0x200;
      p1[0x02]=i&0x00ff; p1[0x03]=i>>8;
      j=(int)((size-0x7c5) / 0x200)+1;
      p1[0x04]=j&0x00ff; p1[0x05]=j>>8;
      lseek(handle,size-0x76E,0); / * move to cs:157H * /
      read(handle,&(p1[0x10]),2); / * read old ss * /
      read(handle,&(p1[0x12]),2); / * read old sp * /
      lseek(handle,0,0);
      if (write(handle,p1,0x18)!=0x18) / * write EXE file-head
* /
        goto end;
      lseek(handle,size-0x7c5,0);
    }
    {
      union REGS r;
      r.h.ah=0x40; r.x.bx=handle;
      r.x.cx=0; intdos(&r,&r);
    }
    removed=1;
  }
  setftime(handle,&filetime);
  end:
    close(handle);
    if (set fattr) chmod(filename,1,flk->ff attrib);
    return removed;
}
int checkmemo() {
  unsigned int seg,off;
  seg=0; off=0x86; seg=peek(seg,off);
  off=0x8c3; off=peek(seg,off);
  if (off==0x1989) return 1;
  else return 0;
}
```