I miss LSD

Tim Brown – Head Of Research

CONTENT/TOPICS

- Why the title?
  - AIX, BSD, HP-UX, IRIX, JVM, Linux, SCO, Solaris
  - Argus Pitbull - http://www.lsd-pl.net/projects/kernvuln.zip
- What this talk will cover
  - So far I've researched...
  - What next?
  - The fascination with AIX

WHY THIS IS IMPORTANT

- UNIX is everywhere
- … except the Desktop (maybe next year?)
  - Android
  - iOS
  - Your favorite embedded device
  - Your bank

THE ATTACK SURFACE

syscall()

- int syscall(int number, ...);
- Kernel provided basic functions (usually)
- Not always kernel land e.g. QNX:
  - Uses MsgSendnc() to send to procmgr
  - We can send from user land
  - Could be a fun target for fuzzing?

MY 1ST KERNEL BUG

- Consider the following:

```
int randomsyscall(int size, void *value) {
        if (size > maxsize) size = maxsize;
        void *a = kmalloc(size);
        privop(a, size);
        copytouser(a, value, size);
        return SUCCESS;
}
```

- What's wrong?

MY 1ST KERNEL BUG

- How about now?

```
int randomsyscall(int size, void *value) {
       if (size > maxsize) size = maxsize;
       void *a = kmalloc(size);
       privop(a, size);
       copytouser(a, value, size);
       return SUCCESS;
}
```

- Signedness bug, impact DoS or disclosure

CVE-2013-2171

- FreeBSD 9.0 vulnerability
- Not mine
- I wrote a PoC for it:
  - mmap() a read-only file
  - ptrace() yourself
  - Write to the location with PIOD_WRITE_D
- Missing permissions check, impact uid=0
- FreeBSD advisories are great for learning

PoC

ioctl()

- int ioctl(int fildes, int request, ... /* arg */);
- Intended to allow device driver interaction
  - Tried my own fuzzer
  - Ported Ilja Van Sprundel's fuzzer
  - return(ENOBUGS)
- But AIX is terrible? Fuzz deeper, “The type of arg depends upon the particular control request, but it shall be either an integer or a pointer to a device-specific data structure.”

IPC

- UNIX sockets
- Signals
- System V semaphores (sem*)
- System V shared memory (shm*)
- System V messages (msg*)
- POSIX semaphores (sem_*)
- POSIX shared memory (shm_*)
- POSIX messages (mq_*)

socket()

- int socket(int domain, int type, int protocol);
- Typically used for establishing TCP and UDP connections
  - There are domains e.g. AF_UNIX
- Scan them with UNIXSocketScan
  - Utilises nmap and custom probes
  - Who knew CUPS supported HTTP over a UNIX socket?
  - How about Avahi's FUCK command?

DEMO

signal()

- void (*signal(int sig, void (*func)(int)))(int);
- Used to trigger functions on exceptional events
- Fuzz them using SIGnalGenerator
- Which signals?
  - Read the source
  - Make use of gdb/objdump
  - USR1 and USR2 are “undefined” - man -K "USR1" - @climagic

REMEMBER LAST YEAR?

- ftp et al are setuid
  - Apparently for additional logging
- Problems? Utilises signals
  - Toggles into a privileged state to log
  - What if we trigger a signal whilst it's uid=0
  - http://lcamtuf.coredump.cx/signals.txt

DEMO

- Not live

shmget()

- int shmget(key_t key, size_t size, int shmflg);
- Patient 0: CVE-2013-025
- Analysing Debian
- Using Coccinelle
- Bugs found?
- Memory corruption?
- What does CERT say?
- Demo
- Status

PATIENT 0: CVE-2013-025

- Spotted by @pentestmonkey
- Insecure permissions on System V Shared Memory
- Affects:
  - QSharedMemoryPrivate
  - QsystemSemaphorePrivate
  - QxcbShmImage
- By extension KDE
- Allows reading and writing

ANALYSING DEBIAN

- Utilised http://codesearch.debian.net/
- (Eventually) Coccinelle – Thanks to grugq!

USING COCCINELLE

```
@shmget@
expression key, size, shmflag;
position p;
@@

shmget@p(key, size, shmflag)

@script:python depends on shmget@
p << shmget.p;
shmflag << shmget.shmflag;
size << shmget.size;
@@
```

USING COCCINELLE

```
if (re.match(".*[0-9][0-9][1-9]([\D]+.*|)$", shmflag) or re.match(".*[0-9][1-9]
    [0-9]([\D]+.*|)$", shmflag) or re.match(".*S_I.(GRP|OTH).*", shmflag)):
        if (re.match(".*bytes_per_line.*", size)):
                print "%s:%s: dangerous shmget(): %s (used for X)" %
                      (p[0].file, p[0].line, shmflag)
        else:
                print "%s:%s: dangerous shmget(): %s" % (p[0].file,
                      p[0].line, shmflag)
elif (re.match(".*[a-z_]+[a-z_]+.*", shmflag)):
        if (re.match(".*bytes_per_line.*", size)):
                print "%s:%s: potentially dangerous shmget(): %s (used for
                      X)" % (p[0].file, p[0].line, shmflag)
        else:
                print "%s:%s: potentially dangerous shmget(): %s" %
                      (p[0].file, p[0].line, shmflag)
```

BUGS FOUND?

- 486 packages using shmget():
  - 89 cases of shmget() being called insecurely, to support X11 protocol (58 packages)
  - 212 other cases of shmget() being called insecurely (114 packages)
  - 80 cases of shmget() being called potentially insecurely, to support X11 (44 packages)
- Similar for semget(), shmctl() and semctl()

MEMORY CORRUPTION?

- Write honoured on both AIX and Linux
- Execute honoured on neither AIX or Linux
  - Always executable on AIX, never on Linux
- ASLR on Linux when randomize_va_space >= 1
- Look in /proc/<pid>/maps for mappings such as "/SYSV00000000 (deleted)”

WHAT DOES CERT SAY?

- Lots of good, generic points around memory management
- “No results found for shmget()”
- On temporary files: “Use other low-level IPC (interprocess communication) mechanisms such as sockets or shared memory”

STATUS

- Qt Project patched generic APIs (CVE-2013-0254)
- Oracle patched Java JRE (CVE-2013-1500)
- Google patched Chrome independently
- No progress made on more general problem with either Red Hat or Debian in almost a year :(

DEMO

shm_open()

- int shm_open(const char *name, int oflag, mode_t mode);
- 0day
  - Allows privilege escalation – more interesting than System V Shared Memory
  - Not on AIX though :P
- Bugs found?

IN PRACTICE

```
17c3ae0:    ba b6 01 00 00        mov   $0x1b6,%edx; mode = 0666
17c3ae5:    be 42 00 00 00        mov   $0x42,%esi; oflag = O_CREAT | O_RDWR – missing O_EXCL
17c3aea:    4c 89 ef              mov   %r13,%rdi
17c3aed:    e8 76 50 a4 fe        callq  208b68 <shm_open@plt>
17c3af2:    45 31 e4              xor   %r12d,%r12d
17c3af5:    83 f8 ff              cmp   $0xffffffff,%eax
17c3af8:    89 c5                 mov   %eax,%ebp
17c3afa:    74 c1                 je    17c3abd
<pthread_attr_setdetachstate@plt+0x15ba3b5>
17c3afc:    be b6 01 00 00        mov   $0x1b6,%esi; mode = 0666
17c3b01:    89 c7                 mov   %eax,%edi
17c3b03:    e8 50 55 a4 fe        callq  209058 <fchmod@plt>
17c3b08:    48 89 de              mov   %rbx,%rsi
17c3b0b:    89 ef                 mov   %ebp,%edi
17c3b0d:    e8 56 59 a4 fe        callq  209468 <ftruncate@plt>
```

fclose(presentation)

Twitter: @portcullislabs

Web: http://labs.portcullis.co.uk/