Linux seccomp(2) vs OpenBSD pledge(2)

Giovanni Bechis
<giovanni@openbsd.org>

Open Source Summit Europe 2017, Prague

SNB Information Technology & Web Solutions

About Me

- sys admin and developer @SNB
- OpenBSD hacker for $\sim$ 10 years
- random patches in random open source software (amavisd-new, courier-imap, cyrus-sasl, memcached, ...)

Mitigations

- stack protector
  - stack buffer underrun detection
- ASLR (Address space layout randomization)
  - prevents exploitation of memory corruption vulnerabilities
- W^X
- priv-sep and priv-drop

dangerous system calls

Disable system calls a process should not call

- SE Linux
- systrace
- Capsicum
- seccomp(2)
- pledge(2)

Linux seccomp(2)

- first version in Linux 2.6.12 (2005)
  - filters enabled via /proc/$PID/seccomp
- in Linux 3.5 (2012) "filter" mode has been added ("seccomp2")
  - can control which syscall are permitted via a BPF "program"
  - some programs are using seccomp(2) (Chrome, OpenSSH, vsftpd, systemd, Firefox, Docker, LXC, ...)
- in Linux 3.8 (2013) via /proc/$PID/status we can obtain the seccomp(2) status (if, for example, seccomp(2) is disabled)
- in Linux 3.17 (2014) seccomp(2) system call has been added as a superset of prctl(2)

OpenBSD pledge(2)

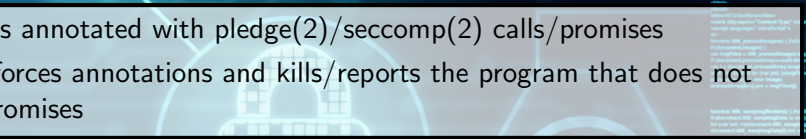
- introduced in OpenBSD 5.8 (2015) as tame(2), than renamed to pledge(2)
- still under development, some new features are coming
- in OpenBSD pledge(2) cannot be disabled
- around 500 programs with pledge(2) support in base
- around 50 ports patched to have pledge(2) support (unzip, mutt, memcached, chromium, ...)

seccomp(2) vs pledge(2)

- approaching the "syscalls" problem
  - study the program
  - figure out all syscalls the program needs
  - "promise" only the operations that are really needed
  - strace(1) or ktrace(1) and gdb(1) if something goes wrong

seccomp(2) vs pledge(2)

- program is annotated with pledge(2)/seccomp(2) calls/promises
- kernel enforces annotations and kills/reports the program that does not respect promises



seccomp(2) vs pledge(2)

```
#include <linux/seccomp.h>
#include <linux/filter.h>
#include <linux/audit.h>
#include <linux/signal.h>
#include <sys/ptrace.h>

int seccomp(unsigned int operation, unsigned int flags, void *args);
```

```
#include <unistd.h>

int pledge(const char *promises, const char *paths[]);
```

seccomp(2) vs pledge(2)

```
$ wc -l seccomp/hello-seccomp.c seccomp/seccomp-bpf.h
  58 seccomp/hello-seccomp.c
  34 seccomp/seccomp-bpf.h
  92 total
```

```
$ wc -l pledge/hello-pledge.c
  17 pledge/hello-pledge.c
```

seccomp(2) vs pledge(2)

```
#define _GNU_SOURCE 1
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <unistd.h>
[...]
static int install_syscall_filter(void)
{
    struct sock_filter filter[] = {
        VALIDATE_ARCHITECTURE,
        EXAMINE_SYSCALL,
[...]
        KILL_PROCESS,
    };
    struct sock_fprog prog = {
        .len = (unsigned short)(sizeof(filter)/sizeof(filter[0])),
        .filter = filter,
    };
    if (prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0)) {
        perror("prctl(NO_NEW_PRIVS)");
        goto failed;
    }
    if (prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, &prog)) {
        perror("prctl(SECCOMP)");
        goto failed;
    }
    return 0;
}
int main(int argc, char **argv) {
FILE *fd;
if((install_syscall_filter()) == 0) {
    printf("Hello !\n");
    if((fd = fopen("/etc/passwd", "r"))) {
        printf("Passwd file opened\n");
    }
    fclose(fd);
    return 0;
} else {
    return 1;
}
```

seccomp(2) vs pledge(2)

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

int main(int argc, char **argv) {
FILE *fd;

if(pledge("stdio", NULL) != -1) {
        printf("Hello !\n");
        if((fd = fopen("/etc/passwd", "r"))) {
                printf("Passwd file opened\n");
        }
        fclose(fd);
        return 0;
} else {
        return 1;
}
}
```

seccomp(2) permitted syscalls

```
#include <linux/seccomp.h>
#include <linux/filter.h>
#include <linux/audit.h>
#include <linux/signal.h>
#include <sys/ptrace.h>

int seccomp(unsigned int operation, unsigned int flags, void *args);
```

- every single syscall available could be allowed

pledge(2) promises

```
#include <unistd.h>

int pledge(const char *promises, const char *paths[]);
```

- "": _exit(2)
- stdio: malloc + rw stdio
- rpath, wpath, cpath, tmppath: open files
- fattr: explicit changes to "fd" (chmod & friends)
- unix, inet: open sockets
- dns: dns requests
- route: routing operations
- sendfd: sends file descriptors via sendmsg(2)
- recvfd: receive file descriptors via recvmsg(2)
- getpw: passwd/group file access
- ioctl: small subset of ioctls is permitted
- tty: subset of ioctl for tty operations
- proc: fork(2), vfork(2), kill(2) and other processes related operations
- exec: execve(2) is allowed to create another process which will be unpledged
- settime: allows to set the system time
- pf: allows a subset of ioctl(2) operations on pf(4) device

seccomp(2) reporting

- if you create a "int install_syscall_reporter(void);" function the kernel will report which syscalls you need.

```
kernel4.4$ ./hello-report
Hello !
Looks like you also need syscall: open(2)
```

seccomp(2) reporting

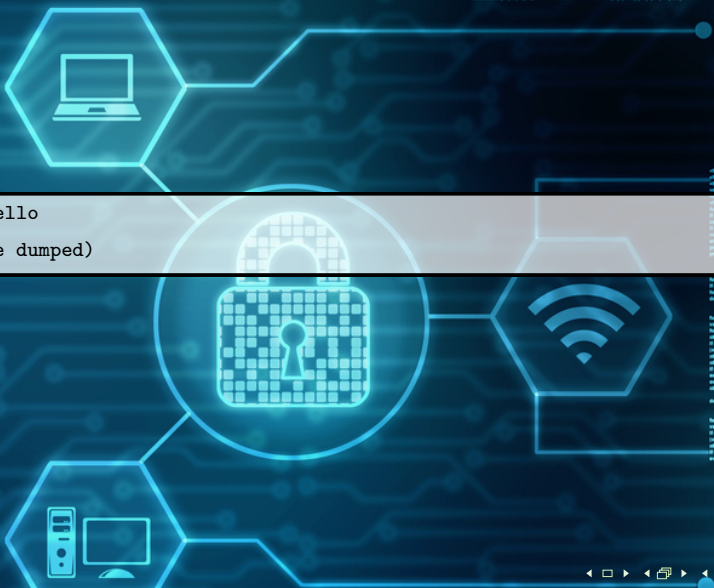
```c
#include "syscall-reporter.h"
#include "syscall-names.h"

const char * const msg_needed = "Looks like you also need syscall: ";

int install_syscall_reporter(void)
{
    struct sigaction act;
    sigset_t mask;
    memset(&act, 0, sizeof(act));
    sigemptyset(&mask);
    sigaddset(&mask, SIGSYS);

    act.sa_sigaction = &reporter;
    act.sa_flags = SA_SIGINFO;
    if (sigaction(SIGSYS, &act, NULL) < 0) {
        perror("sigaction");
        return -1;
    }
    if (sigprocmask(SIG_UNBLOCK, &mask, NULL)) {
        perror("sigprocmask");
        return -1;
    }
    return 0;
}
```

seccomp(2) reporting

```
kernel2.6.32$ ./hello
prctl(NO_NEW_PRIVS): Invalid argument
SECCOMP_FILTER is not available. :(

kernel3.10$ ./hello
Bad system call

kernel3.10$ ./hello-report
Looks like you also need syscall: mmap(9)

kernel4.4$ ./hello
Hello !
Bad system call

kernel4.4$ ./hello-report
Hello !
Looks like you also need syscall: open(2)
```

pledge(2) reporting

```
openbsd6.2$ ./hello
Hello !
Abort trap (core dumped)
```



seccomp(2) logging

- dmesg(1) and rsyslogd(8)

```
Oct 12 16:02:56 ubuntu kernel: auditd: type:1326 audit(1507816976.188:30):
auid=1000 uid=1000 gid=1000 ses=1 subj=system_u:system_r:kernel_t:s0 pid=1227
comm="hello" exe="/home/test/src/hello" sig=31 arch=c000003e syscall=2 compat=0
ip=0x7fbb3ab75010 code=0x0
Oct 12 16:02:56 ubuntu audit[1227]: SECCOMP auid=1000 uid=1000 gid=1000 ses=1
subj=system_u:system_r:kernel_t:s0 pid=1227 comm="hello"
exe="/home/test/src/hello" sig=31 arch=c000003e syscall=2 compat=0
ip=0x7fbb3ab75010 code=0x0
```

pledge(2) logging

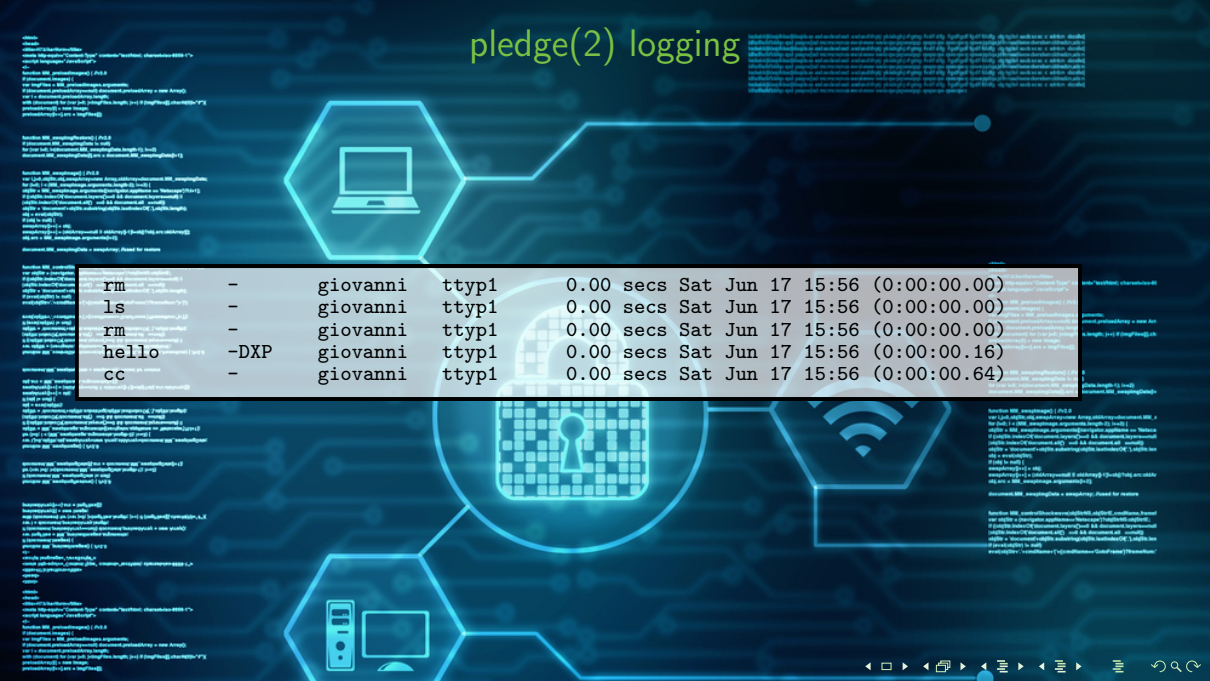
- dmesg(8) and syslogd(8)

```
Oct 12 16:17:30 laptop /bsd: hello(31340): syscall 5 "rpath"
```

```
$ grep -A1 "^5    STD" /usr/src/sys/kern/syscalls.master
5       STD                     { int sys_open(const char *path, \
                                    int flags, ... mode_t mode); }
```

- lastcomm(1) and daily(8) on OpenBSD $\geq$ 6.2 (with accounting enabled)

pledge(2) logging



```
rm        -       giovanni    ttyp1       0.00 secs Sat Jun 17 15:56 (0:00:00.00)
ls        -       giovanni    ttyp1       0.00 secs Sat Jun 17 15:56 (0:00:00.00)
rm        -       giovanni    ttyp1       0.00 secs Sat Jun 17 15:56 (0:00:00.00)
hello     -DXP    giovanni    ttyp1       0.00 secs Sat Jun 17 15:56 (0:00:00.16)
cc        -       giovanni    ttyp1       0.00 secs Sat Jun 17 15:56 (0:00:00.64)
```

pledge(2) logging

```
From root@bigio.paclan.it Sat Jun 17 16:08:46 2017
Delivered-To: root@bigio.paclan.it
From: Charlie Root <root@bigio.paclan.it>
To: root@bigio.paclan.it
Subject: bigio.paclan.it daily output

OpenBSD 6.1-current (GENERIC) #1: Fri Jun 16 22:37:23 CEST 2017
    giovanni@bigio.paclan.it:/usr/src/sys/arch/amd64/compile/GENERIC

 4:08PM  up 35 mins, 3 users, load averages: 0.26, 0.13, 0.10

Purging accounting records:
hello      -DXP    giovanni   ttyp1      0.00 secs Sat Jun 17 15:56 (0:00:00.16)
```

adding pledge(1) support to ∼ 500 programs

```
Index: worms.c
===================================================================
RCS file: /var/cvs/src/games/worms/worms.c,v
retrieving revision 1.22
retrieving revision 1.23
diff -u -p -r1.22 -r1.23
--- worms.c 18 Feb 2015 23:16:08 -0000 1.22
+++ worms.c 21 Nov 2015 05:29:42 -0000 1.23
@@ -1,4 +1,4 @@
-/* $OpenBSD: worms.c,v 1.22 2015/02/18 23:16:08 tedu Exp $ */
+/* $OpenBSD: worms.c,v 1.23 2015/11/21 05:29:42 deraadt Exp $ */

 /*
  * Copyright (c) 1980, 1993
@@ -182,6 +182,9 @@ main(int argc, char *argv[])
    struct termios term;
    speed_t speed;
    time_t delay = 0;
+
+   if (pledge("stdio rpath tty", NULL) == -1)
+       err(1, "pledge");

    /* set default delay based on terminal baud rate */
    if (tcgetattr(STDOUT_FILENO, &term) == 0 &&
```

[memcached] adding seccomp support

```
#include "config.h"
#include <seccomp.h>
#include <errno.h>
#include <stdlib.h>
#include "memcached.h"

// In the future when the system is more tested this could be switched
// to SCMP_ACT_KILL instead.
#define DENY_ACTION SCMP_ACT_ERRNO(EACCES)

void drop_privileges(void) {
    scmp_filter_ctx ctx = seccomp_init(DENY_ACTION);
    if (ctx == NULL) {
        return;
    }
    int rc = 0;
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(sigreturn), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(futex), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(epoll_wait), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(accept4), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(accept), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(write), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(fstat), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(mmap), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(munmap), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(shmctl), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(exit_group), 0);

#ifdef MEMCACHED_DEBUG
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(open), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(fcntl), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(read), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(lseek), 0);
    rc |= seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(close), 0);
#endif
    if (rc != 0) {
        goto fail;
    }
    rc = seccomp_load(ctx);
    if (rc < 0) {
        goto fail;
    }
    [...]
}
```

[memcached] adding pledge support

```
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include "memcached.h"

/*
 * this section of code will drop all (OpenBSD) privileges including
 * those normally granted to all userland process (basic privileges). The
 * effect of this is that after running this code, the process will not able
 * to fork(), exec(), etc.  See pledge(2) for more information.
 */
void drop_privileges() {
    extern char *__progname;

    if (settings.socketpath != NULL) {
        if (pledge("stdio unix", NULL) == -1) {
            fprintf(stderr, "%s: pledge: %s\n", __progname, strerror(errno));
            exit(EXIT_FAILURE);
        }
    } else {
        if (pledge("stdio inet", NULL) == -1) {
            fprintf(stderr, "%s: pledge: %s\n", __progname, strerror(errno));
            exit(EXIT_FAILURE);
        }
    }
}
```

what to do if something goes wrong ?

```
prctl(PR_SET_NO_PRIVS,1, 0, 0, 0) = 0
prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, {len = 19, filter = 0x7fffc3349260}) = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(4, 1), ...}) = 0
ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
brk(NULL)
brk(0xd83000)
write(1, "Hello !\n", 8Hello !
) = 8
+++ Killed by SYGSYS +++
Bad system call (core dumped)
```

what to do if something goes wrong ?

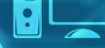
```
94140 hello    CALL  write(1,0xb56246ae000,0x8)
94140 hello    GIO   fd 1 wrote 8 bytes
               "Hello !
               "
94140 hello    RET   write 8
94140 hello    CALL  kbind(0x7f7ffffcbee8,24,0x73b422cd44dee9e4)
94140 hello    RET   kbind 0
94140 hello    CALL  open(0xb53f8a00b20,0<O_RDONLY>)
94140 hello    NAMI  "/etc/passwd"
94140 hello    PLDG  open, "rpath", errno 1 Operation not permitted
94140 hello    PSIG  SIGABRT SIG_DFL
94140 hello    NAMI  "hello.core"
```

what to do if something goes wrong ?

```
$ gdb hello hello.core
GNU gdb 6.3
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "amd64-unknown-openbsd6.1"...
Core was generated by `hello'.
Program terminated with signal 6, Aborted.
Loaded symbols for /home/data/ossummit_2017/src/hello
Reading symbols from /usr/lib/libc.so.89.5...done.
Loaded symbols for /usr/lib/libc.so.89.5
Reading symbols from /usr/libexec/ld.so...done.
Loaded symbols for /usr/libexec/ld.so
#0  0x000009f584a507aa in _thread_sys_open () at {standard input}:5
5       {standard input}: No such file or directory.
        in {standard input}
(gdb) bt
#0  0x000009f584a507aa in _thread_sys_open () at {standard input}:5
#1  0x000009f584a3f559 in *_libc_open_cancel (path=Variable "path" is not available.
)
    at /usr/src/lib/libc/sys/w_open.c:36
#2  0x000009f584aaab82 in *_libc_fopen (file=0x9f2b8b00b20 "/etc/passwd",
 mode=Variable "mode" is not available.

) at /usr/src/lib/libc/stdio/fopen.c:54
#3  0x000009f2b8a005dc in main (argc=1, argv=0x7f7ffffc3c58) at hello.c:8
Current language:  auto; currently asm
(gdb)
```

don't you speak "C" ?

```
import sys, os
from seccomp import *

f = SyscallFilter(defaction=KILL)

f.add_rule(ALLOW, "exit_group")
f.add_rule(ALLOW, "rt_sigaction")
f.add_rule(ALLOW, "brk")
f.add_rule(ALLOW, "open")
f.add_rule(ALLOW, "write", Arg(0, EQ, sys.stdout.fileno()))

f.load()

tmp_fd = os.open('/tmp/test.txt', os.O_WRONLY)
os.write(tmp_fd, 'Hello, world\n')
```



don't you speak "C" ?

```
use OpenBSD::Pledge;
my $file = "/usr/share/dict/words";
pledge(qw( rpath )) || die "Unable to pledge: $!";

open my $fh, '<', $file or die "Unable to open $file: $!\n";
while ( readline($fh) ) {
    print if /pledge/i;
}
close $fh;
system("ls");
```

The future ?

seccomp(2) vs pledge(2) - Bibliography

- http://man7.org/linux/man-pages/man2/seccomp.2.html
- https://man.openbsd.org/pledge.2
- https://wiki.mozilla.org/Security/Sandbox/Seccomp
- https://github.com/aggsol/linux-pledge
- https://github.com/seccomp/libseccomp
- https://github.com/afresh1/OpenBSD-Pledge