```
**********************************************************
    5793 Mon Aug 29 10:16:14 2016
new/usr/src/man/man1/psecflags.1
sync further changes from uts/aslr
**********************************************************
'\" te
.\" This file and its contents are supplied under the terms of the
.\" Common Development and Distribution License ("CDDL"), version 1.0.
.\" You may only use this file in accordance with the terms of version
.\" 1.0 of the CDDL.
.\"
.\" A full copy of the text of the CDDL should have accompanied this
.\" source.  A copy of the CDDL is also available via the Internet at
.\" http://www.illumos.org/license/CDDL.
.\"
.\" Copyright 2015, Richard Lowe.
.\"
.TH "PSECFLAGS" "1" "June 6, 2016"
.SH "NAME"
\fBpsecflags\fR - inspect or modify process security flags
.SH "SYNOPSIS"
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-s\fR \fIspec\fR \fI-e\fR \fIcommand\fR \
[\fIarg\fR]...
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-s\fR \fIspec\fR [\fI-i\fR \fIidtype\fR] \
\fIid\fR ...
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR [\fI-F\fR] { \fIpid\fR | \fIcore\fR }
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-l\fR
.fi

.SH "DESCRIPTION"
The first invocation of the \fBpsecflags\fR command runs the specified
\fIcommand\fR with the security-flags modified as described by the \fI-s\fR
argument.
.P
The second invocation modifies the security-flags of the processes described
by \fIidtype\fR and \fIid\fR according as described by the \fI-s\fR argument.
.P
The third invocation describes the security-flags of the specified processes
or core files.  The effective set is signified by '\fBE\fR', the inheritable
set by '\fBI\fR', the lower set by '\fBL\fR', and the upper set by '\fBU\fR'.
.P
The fourth invocation lists the supported process security-flags, documented
in \fBsecurity-flags\fR(5).

.SH "OPTIONS"
The following options are supported:
.sp
.ne 2
.na
\fB-e\fR
.ad
.RS 11n
Interpret the remaining arguments as a command line and run the command with
the security-flags specified with the \fI-s\fR flag.
.RE
```

```
.sp
.ne 2
.na
\fB-F\fR
.ad
.RS 11n
Force. Grab the target process even if another process has control.
.RE

.sp
.ne 2
.na
\fB-i\fR \fIidtype\fR
.ad
.RS 11n
This option, together with the \fIid\fR arguments specify one or more
processes whose security-flags will be modified. The interpretation of the
\fIid\fR arguments is based on \fIidtype\fR. If \fIidtype\fR is omitted the
default is \fBpid\fR.

Valid \fIidtype\fR options are:
.sp
.ne 2
.na
\fBall\fR
.ad
.RS 11n
The \fBpsecflags\fR command applies to all processes
.RE

.sp
.ne 2
.na
\fBcontract\fR, \fBctid\fR
.ad
.RS 11n
The security-flags of any process with a contract ID matching the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBgroup\fR, \fBgid\fR
.ad
.RS 11n
The security-flags of any process with a group ID matching the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBpid\fR
.ad
.RS 11n
The security-flags of any process with a process ID matching the \fIid\fR
arguments are modified. This is the default.
.RE

.sp
.ne 2
.na
\fBppid\fR
.ad
```

```
.RS 11n
The security-flags of any processes whose parent process ID matches the
\fIid\fR arguments are modified.
.RE

.sp
.ne 2
.na
\fBproject\fR, \fBprojid\fR
.ad
.RS 11n
The security-flags of any process whose project ID matches the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBsession\fR, \fBsid\fR
.ad
.RS 11n
The security-flags of any process whose session ID matches the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBtaskid\fR
.ad
.RS 11n
The security-flags of any process whose task ID matches the \fIid\fR arguments
are modified.
.RE

.sp
.ne 2
.na
\fBuser\fR, \fBuid\fR
.ad
.RS 11n
The security-flags of any process belonging to the users matching the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBzone\fR, \fBzoneid\fR
.ad
.RS 11n
The security-flags of any process running in the zones matching the given
\fIid\fR arguments are modified.
.RE
.RE

.sp
.ne 2
.na
\fB-l\fR
.ad
.RS 11n
List all supported process security-flags, described in
\fBsecurity-flags\fR(5).
.RE
```

```
.sp
.ne 2
.na
\fB-s\fR \fIspecification\fR
.ad
.RS 11n
Modify the process security-flags according to
\fIspecification\fR. Specifications take the form of a comma-separated list of
flags, optionally preceded by a '-' or '!'. Where '-' and '!' indicate that the
given flag should be removed from the specification.  The pseudo-flags "all",
"none" and "current" are supported, to indicate that all flags, no flags, or
the current set of flags (respectively) are to be included.
.P
By default, the inheritable flags are changed.  You may optionally specify the
set to change using their single-letter identifiers and an equals sign.
.P
For a list of valid security-flags, see \fBpsecflags -l\fR.
.RE

.SH "EXAMPLES"
.LP
\fBExample 1\fR Display the security-flags of the current shell.
.sp
.in +2
.nf
example$ \fBpsecflags $$\fR
100718: -sh
        E:      aslr
        I:      aslr
        L:      none
        U:      aslr,forbidnullmap,noexecstack
.fi
.in -2
.sp

.LP
\fBExample 2\fR Run a user command with ASLR enabled in addition to any
inherited security flags.
.sp
.in +2
.nf
example$ \fBpsecflags -s current,aslr -e /bin/sh\fR
$ psecflags $$
100724: -sh
        E:      none
        I:      aslr
        L:      none
        U:      aslr,forbidnullmap,noexecstack
.fi
.in -2
.sp

.LP
\fBExample 3\fR Remove aslr from the inheritable flags of all Bob's processes.
.sp
.in +2
.nf
example# \fBpsecflags -s current,-aslr -i uid bob\fR
.fi
.in -2

.LP
\fBExample 4\fR Add the aslr flag to the lower set, so that all future
child processes must have this flag set.
.sp
.in +2
```

```
.nf
example# \fBpsecflags -s L=current,aslr $$\fR
.fi
.in -2

.SH "EXIT STATUS"
The following exit values are returned:

.TP
\fB0\fR
.IP
Success.

.TP
\fBnon-zero\fR
.IP
An error has occurred.
An error has occured.

.SH "ATTRIBUTES"
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Volatile
.TE

.SH "SEE ALSO"
.BR exec (2),
.BR attributes (5),
.BR contract (4),
.BR security-flags (5),
.BR zones (5)
```

****
1301 Mon Aug 29 10:16:16 2016
new/usr/src/test/os-tests/tests/secflags/secflags_core.sh
sync further changes from uts/aslr
****
______unchanged_portion_omitted_

```
trap cleanup EXIT

## gcore-produced core
gcore $pid >/dev/null

cat > gcore-expected.$$ <<EOF
core 'core.$pid' of $pid:       sleep 100000
        E:      aslr
        I:      aslr
EOF

/usr/bin/psecflags core.${pid} | grep -v '[LU]:' > gcore-output.$$

if ! diff -u gcore-expected.$$ gcore-output.$$; then
    exit 1;
fi

## kernel-produced core
kill -SEGV $pid
wait $pid >/dev/null 2>&1
#endif /* ! codereview */

cat > core-expected.$$ <<EOF
core 'core' of $pid:    sleep 100000
        E:      aslr
        I:      aslr
EOF

/usr/bin/psecflags core | grep -v '[LU]:' > core-output.$$

if ! diff -u core-expected.$$ core-output.$$; then
    exit 1;
fi

exit 0
```

```
**********************************************************
    1728 Mon Aug 29 10:16:17 2016
new/usr/src/test/os-tests/tests/secflags/secflags_dts.sh
sync further changes from uts/aslr
**********************************************************
______unchanged_portion_omitted_
EOF

gcc -o tester-aslr tester.c -Wl,-z,aslr=enabled
gcc -o tester-noaslr tester.c -Wl,-z,aslr=disabled

# This is the easiest way I've found to get many many DTs, but it's gross
gcc -o many-dts-aslr tester.c -Wl,-z,aslr=enabled $(for elt in /usr/lib/lib*.so;
gcc -o many-dts-noaslr tester.c -Wl,-z,aslr=disabled $(for elt in /usr/lib/lib*.

check() {
    bin=$1
    state=$2
    set=$3
#endif /* ! codereview */
    ret=0

    $bin &
    pid=$!
    psecflags $pid | grep -q "${set}:.*aslr"
    psecflags $pid | grep -q 'E:.*aslr'
    (( $? != $state )) && ret=1
    kill -9 $pid
    return $ret
}
______unchanged_portion_omitted_

psecflags -s none $$
check ./tester-aslr 0 E || fail "DT_SUNW_ASLR 1 failed"
check ./many-dts-aslr 0 E || fail "DT_SUNW_ASLR 1 with many DTs failed"
check ./tester-aslr 1 I || fail "DT_SUNW_ASLR 1 incorrectly set the inheritable
check ./tester-aslr 0 || fail "DT_SUNW_ASLR 1 failed"
check ./many-dts-aslr 0 || fail "DT_SUNW_ASLR 1 with many DTs failed"

psecflags -s aslr $$
check ./tester-noaslr 1 E || fail "DT_SUNW_ASLR 0 failed"
check ./many-dts-noaslr 1 E || fail "DT_SUNW_ASLR 0 with many DTs failed"
check ./tester-noaslr 1 || fail "DT_SUNW_ASLR 0 failed"
check ./many-dts-noaslr 1 || fail "DT_SUNW_ASLR 0 with many DTs failed"
```

****

1780 Mon Aug 29 10:16:18 2016
new/usr/src/test/os-tests/tests/secflags/secflags_elfdump.sh
sync further changes from uts/aslr

****
______unchanged_portion_omitted_

```
trap cleanup EXIT

## gcore-produced core
gcore $pid >/dev/null

cat > gcore-expected.$$ <<EOF
    namesz: 0x5
    descsz: 0x28
    type:   [ NT_SECFLAGS ]
    name:
        CORE\0
    desc: (prsecflags_t)
        pr_version:    1
        pr_effective:  [ ASLR ]
        pr_inherit:    [ ASLR ]
        pr_lower:      0
        pr_upper:      [ ASLR FORBIDNULLMAP NOEXECSTACK ]
EOF

/usr/bin/elfdump -n core.${pid} | grep -B5 -A5 prsecflags_t > gcore-output.$$

if ! diff -u gcore-expected.$$ gcore-output.$$; then
    exit 1;
fi

## kernel-produced core
kill -SEGV $pid
wait $pid >/dev/null 2>&1
#endif /* ! codereview */

cat > core-expected.$$ <<EOF
    namesz: 0x5
    descsz: 0x28
    type:   [ NT_SECFLAGS ]
    name:
        CORE\0
    desc: (prsecflags_t)
        pr_version:    1
        pr_effective:  [ ASLR ]
        pr_inherit:    [ ASLR ]
        pr_lower:      0
        pr_upper:      [ ASLR FORBIDNULLMAP NOEXECSTACK ]
EOF

/usr/bin/elfdump -n core | grep -B5 -A5 prsecflags_t > core-output.$$

if ! diff -u core-expected.$$ core-output.$$; then
    exit 1;
fi

exit 0
```

****
59159 Mon Aug 29 10:16:19 2016
new/usr/src/uts/common/exec/elf/elf.c
sync further changes from uts/aslr
****
______***unchanged_portion_omitted_***

```
static int
handle_secflag_dt(proc_t *p, uint_t dt, uint_t val)
{
        uint_t flag;

        switch (dt) {
        case DT_SUNW_ASLR:
                flag = PROC_SEC_ASLR;
                break;
        default:
                return (EINVAL);
        }

        if (val == 0) {
                if (secflag_isset(p->p_secflags.psf_lower, flag))
                        return (EPERM);
                if ((secpolicy_psecflags(CRED(), p, p) != 0) &&
                    secflag_isset(p->p_secflags.psf_inherit, flag))
                        return (EPERM);

                secflag_clear(&p->p_secflags.psf_inherit, flag);
                secflag_clear(&p->p_secflags.psf_effective, flag);
        } else {
                if (!secflag_isset(p->p_secflags.psf_upper, flag))
                        return (EPERM);

                if ((secpolicy_psecflags(CRED(), p, p) != 0) &&
                    !secflag_isset(p->p_secflags.psf_inherit, flag))
                        return (EPERM);

                secflag_set(&p->p_secflags.psf_inherit, flag);
                secflag_set(&p->p_secflags.psf_effective, flag);
        }

        return (0);
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
   26807 Mon Aug 29 10:16:20 2016
new/usr/src/uts/common/os/grow.c
sync further changes from uts/aslr
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/* Copyright 2013 OmniTI Computer Consulting, Inc. All rights reserved. */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */

#include <sys/types.h>
#include <sys/inttypes.h>
#include <sys/param.h>
#include <sys/sysmacros.h>
#include <sys/systm.h>
#include <sys/signal.h>
#include <sys/user.h>
#include <sys/errno.h>
#include <sys/var.h>
#include <sys/proc.h>
#include <sys/tuneable.h>
#include <sys/debug.h>
#include <sys/cmn_err.h>
#include <sys/cred.h>
#include <sys/vnode.h>
#include <sys/vfs.h>
#include <sys/vm.h>
#include <sys/file.h>
#include <sys/mman.h>
#include <sys/vmparam.h>
#include <sys/fcntl.h>
#include <sys/lwpchan_impl.h>
#include <sys/nbmlock.h>

#include <vm/hat.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/seg_dev.h>
#include <vm/seg_vn.h>
```

```
int use_brk_lpg = 1;
int use_stk_lpg = 1;

/*
 * If set, we will not randomize mappings where the 'addr' argument is
 * non-NULL and not an alignment.
 */
int aslr_respect_mmap_hint = 0;

#endif /* ! codereview */
static int brk_lpg(caddr_t nva);
static int grow_lpg(caddr_t sp);

intptr_t
brk(caddr_t nva)
{
        int error;
        proc_t *p = curproc;

        /*
         * Serialize brk operations on an address space.
         * This also serves as the lock protecting p_brksize
         * and p_brkpageszc.
         */
        as_rangelock(p->p_as);

        /*
#endif /* ! codereview */
         * As a special case to aid the implementation of sbrk(3C), if given a
         * new brk of 0, return the current brk.  We'll hide this in brk(3C).
         */
        if (nva == 0) {
                as_rangeunlock(p->p_as);
        if (nva == 0)
                return ((intptr_t)(p->p_brkbase + p->p_brksize));
        }
#endif /* ! codereview */

        /*
         * Serialize brk operations on an address space.
         * This also serves as the lock protecting p_brksize
         * and p_brkpageszc.
         */
        as_rangelock(p->p_as);
        if (use_brk_lpg && (p->p_flag & SAUTOLPG) != 0) {
                error = brk_lpg(nva);
        } else {
                error = brk_internal(nva, p->p_brkpageszc);
        }
        as_rangeunlock(p->p_as);
        return ((error != 0 ? set_errno(error) : 0));
}
______unchanged_portion_omitted_

#define RANDOMIZABLE_MAPPING(addr, flags) (((flags & MAP_FIXED) == 0) && \
        !(((flags & MAP_ALIGN) == 0) && (addr != 0) && aslr_respect_mmap_hint))

#endif /* ! codereview */
static int
smmap_common(caddr_t *addrp, size_t len,
    int prot, int flags, struct file *fp, offset_t pos)
{
        struct vnode *vp;
        struct as *as = curproc->p_as;
        uint_t uprot, maxprot, type;
        int error;
```

```c
        int in_crit = 0;

        if ((flags & ~(MAP_SHARED | MAP_PRIVATE | MAP_FIXED | _MAP_NEW |
            _MAP_LOW32 | MAP_NORESERVE | MAP_ANON | MAP_ALIGN |
            MAP_TEXT | MAP_INITDATA)) != 0) {
                /* | MAP_RENAME */      /* not implemented, let user know */
                return (EINVAL);
        }

        if ((flags & MAP_TEXT) && !(prot & PROT_EXEC)) {
                return (EINVAL);
        }

        if ((flags & (MAP_TEXT | MAP_INITDATA)) == (MAP_TEXT | MAP_INITDATA)) {
                return (EINVAL);
        }

        if ((flags & (MAP_FIXED | _MAP_RANDOMIZE)) ==
            (MAP_FIXED | _MAP_RANDOMIZE)) {
                return (EINVAL);
        }

        /*
         * If it's not a fixed allocation and mmap ASLR is enabled, randomize
         * it.
         */
        if (RANDOMIZABLE_MAPPING(*addrp, flags) &&
        if (((flags & MAP_FIXED) == 0) &&
            secflag_enabled(curproc, PROC_SEC_ASLR))
                flags |= _MAP_RANDOMIZE;

#if defined(__sparc)
        /*
         * See if this is an "old mmap call".  If so, remember this
         * fact and convert the flags value given to mmap to indicate
         * the specified address in the system call must be used.
         * _MAP_NEW is turned set by all new uses of mmap.
         */
        if ((flags & _MAP_NEW) == 0)
                flags |= MAP_FIXED;
#endif
        flags &= ~_MAP_NEW;

        type = flags & MAP_TYPE;
        if (type != MAP_PRIVATE && type != MAP_SHARED)
                return (EINVAL);


        if (flags & MAP_ALIGN) {
                if (flags & MAP_FIXED)
                        return (EINVAL);

                /* alignment needs to be a power of 2 >= page size */
                if (((uintptr_t)*addrp < PAGESIZE && (uintptr_t)*addrp != 0) ||
                    !ISP2((uintptr_t)*addrp))
                        return (EINVAL);
        }
        /*
         * Check for bad lengths and file position.
         * We let the VOP_MAP routine check for negative lengths
         * since on some vnode types this might be appropriate.
         */
        if (len == 0 || (pos & (u_offset_t)PAGEOFFSET) != 0)
                return (EINVAL);

        maxprot = PROT_ALL;             /* start out allowing all accesses */
```

```c
        uprot = prot | PROT_USER;

        if (fp == NULL) {
                ASSERT(flags & MAP_ANON);
                /* discard lwpchan mappings, like munmap() */
                if ((flags & MAP_FIXED) && curproc->p_lcp != NULL)
                        lwpchan_delete_mapping(curproc, *addrp, *addrp + len);
                as_rangelock(as);
                error = zmap(as, addrp, len, uprot, flags, pos);
                as_rangeunlock(as);
                /*
                 * Tell machine specific code that lwp has mapped shared memory
                 */
                if (error == 0 && (flags & MAP_SHARED)) {
                        /* EMPTY */
                        LWP_MMODEL_SHARED_AS(*addrp, len);
                }
                return (error);
        } else if ((flags & MAP_ANON) != 0)
                return (EINVAL);

        vp = fp->f_vnode;

        /* Can't execute code from "noexec" mounted filesystem. */
        if ((vp->v_vfsp->vfs_flag & VFS_NOEXEC) != 0)
                maxprot &= ~PROT_EXEC;

        /*
         * These checks were added as part of large files.
         *
         * Return ENXIO if the initial position is negative; return EOVERFLOW
         * if (offset + len) would overflow the maximum allowed offset for the
         * type of file descriptor being used.
         */
        if (vp->v_type == VREG) {
                if (pos < 0)
                        return (ENXIO);
                if ((offset_t)len > (OFFSET_MAX(fp) - pos))
                        return (EOVERFLOW);
        }

        if (type == MAP_SHARED && (fp->f_flag & FWRITE) == 0) {
                /* no write access allowed */
                maxprot &= ~PROT_WRITE;
        }

        /*
         * XXX - Do we also adjust maxprot based on protections
         * of the vnode?  E.g. if no execute permission is given
         * on the vnode for the current user, maxprot probably
         * should disallow PROT_EXEC also?  This is different
         * from the write access as this would be a per vnode
         * test as opposed to a per fd test for writability.
         */

        /*
         * Verify that the specified protections are not greater than
         * the maximum allowable protections.  Also test to make sure
         * that the file descriptor does allows for read access since
         * "write only" mappings are hard to do since normally we do
         * the read from the file before the page can be written.
         */
        if (((maxprot & uprot) != uprot) || (fp->f_flag & FREAD) == 0)
                return (EACCES);

        /*
```

```
         * If the user specified an address, do some simple checks here
         */
        if ((flags & MAP_FIXED) != 0) {
                caddr_t userlimit;

                /*
                 * Use the user address.  First verify that
                 * the address to be used is page aligned.
                 * Then make some simple bounds checks.
                 */
                if (((uintptr_t)*addrp & PAGEOFFSET) != 0)
                        return (EINVAL);

                userlimit = flags & _MAP_LOW32 ?
                    (caddr_t)USERLIMIT32 : as->a_userlimit;
                switch (valid_usr_range(*addrp, len, uprot, as, userlimit)) {
                case RANGE_OKAY:
                        break;
                case RANGE_BADPROT:
                        return (ENOTSUP);
                case RANGE_BADADDR:
                default:
                        return (ENOMEM);
                }
        }

        if ((prot & (PROT_READ | PROT_WRITE | PROT_EXEC)) &&
            nbl_need_check(vp)) {
                int svmand;
                nbl_op_t nop;

                nbl_start_crit(vp, RW_READER);
                in_crit = 1;
                error = nbl_svmand(vp, fp->f_cred, &svmand);
                if (error != 0)
                        goto done;
                if ((prot & PROT_WRITE) && (type == MAP_SHARED)) {
                        if (prot & (PROT_READ | PROT_EXEC)) {
                                nop = NBL_READWRITE;
                        } else {
                                nop = NBL_WRITE;
                        }
                } else {
                        nop = NBL_READ;
                }
                if (nbl_conflict(vp, nop, 0, LONG_MAX, svmand, NULL)) {
                        error = EACCES;
                        goto done;
                }
        }

        /* discard lwpchan mappings, like munmap() */
        if ((flags & MAP_FIXED) && curproc->p_lcp != NULL)
                lwpchan_delete_mapping(curproc, *addrp, *addrp + len);

        /*
         * Ok, now let the vnode map routine do its thing to set things up.
         */
        error = VOP_MAP(vp, pos, as,
            addrp, len, uprot, maxprot, flags, fp->f_cred, NULL);

        if (error == 0) {
                /*
                 * Tell machine specific code that lwp has mapped shared memory
                 */
                if (flags & MAP_SHARED) {
```

```
                        /* EMPTY */
                        LWP_MMODEL_SHARED_AS(*addrp, len);
                }
                if (vp->v_type == VREG &&
                    (flags & (MAP_TEXT | MAP_INITDATA)) != 0) {
                        /*
                         * Mark this as an executable vnode
                         */
                        mutex_enter(&vp->v_lock);
                        vp->v_flag |= VVMEXEC;
                        mutex_exit(&vp->v_lock);
                }
        }

done:
        if (in_crit)
                nbl_end_crit(vp);
        return (error);
}
```

______***unchanged_portion_omitted_***