Capsicum and Casper

Paweł Jakub Dawidek
<pjd@FreeBSD.org>

System capabilities

```
int cap_enter(void)
int cap_getmode(u_int *modep)

int cap_rights_limit(int fd, const cap_rights_t *rightsp)
int cap_rights_get(int fd, cap_rights_t *rightsp)

int cap_ioctls_limit(int fd, const unsigned long *cmds, size_t ncmds)
int cap_ioctls_get(int fd, unsigned long *cmds, size_t maxcmds)

int cap_fcntls_limit(int fd, uint32_t fcntlrights)
int cap_fcntls_get(int fd, uint32_t *fcntlrightsp)
```

Capability rights for system capabilities

CAP_ACCEPT
CAP_BIND
CAP_FCHMOD
CAP_FCHOWN
CAP_IOCTL
CAP_SEEK
CAP_READ
CAP_WRITE

From uint64_t to struct cap_rights

In 10-CURRENT:

```
typedef uint64_t        cap_rights_t;
```

In pjd_capsicum:

```
struct cap_rights {
        uint64_t   cr_rights[CAP_RIGHTS_VERSION + 2];
};
typedef struct cap_rights        cap_rights_t;
```

From uint64_t to struct cap_rights

In 10-CURRENT:

```
#define CAP_READ      0x0000000000000001ULL
#define CAP_BINDAT    0x0400000000000000ULL
```

In pjd_capsicum:

```
#define CAPRIGHT(idx, bit)    ((1ULL << (57 + (idx))) | (bit))

#define CAP_READ      CAPRIGHT(0, 0x0000000000000001ULL)
#define CAP_BINDAT    CAPRIGHT(1, 0x0000000000001000ULL)
```

Managing new cap_rights_t

```
cap_rights_t *cap_rights_init(cap_rights_t *rights, ...)

void cap_rights_set(cap_rights_t *rights, ...)
void cap_rights_clear(cap_rights_t *rights, ...)
bool cap_rights_is_set(const cap_rights_t *rights, ...)

bool cap_rights_is_valid(const cap_rights_t *rights)
void cap_rights_merge(cap_rights_t *dst, const cap_rights_t *src)
void cap_rights_remove(cap_rights_t *dst, const cap_rights_t *src)
void cap_rights_contains(const cap_rights_t *big, const cap_rights_t *little)
```

Code changes

In 10-CURRENT:

```
error = getvnode(fdp, uap->fd, CAP_FCHDIR, &fp);
```

In pjd_capsicum:

```
cap_rights_t rights;

error = getvnode(fdp, uap->fd, cap_rights_init(&rights, CAP_FCHDIR), &fp);
```

Code changes (10-CURRENT)

```
cap_rights_t rights;
unsigned long cmd;

rights = CAP_FSTAT;
cmd = -1;

switch (fd) {
case STDIN_FILENO:
	rights |= CAP_READ;
	break;
case STDOUT_FILENO:
	rights |= CAP_IOCTL | CAP_WRITE;
	cmd = TIOCGETA; /* required by isatty(3) in printf(3) */
	break;
case STDERR_FILENO:
	rights |= CAP_WRITE;
	if (!suppressdata) {
		rights |= CAP_IOCTL;
		cmd = TIOCGWINSZ;
	}
	break;
default:
	abort();
}

if (cap_rights_limit(fd, rights) < 0 && errno != ENOSYS)
	err(1, "unable to limit rights for descriptor %d", fd);
if (cap_ioctls_limit(fd, &cmd, 1) < 0 && errno != ENOSYS)
	err(1, "unable to limit ioctls for descriptor %d", fd);
```

Code changes (pjd_capsicum)

```
cap_rights_t rights;
unsigned long cmd;

cap_rights_init(&rights, CAP_FSTAT);
cmd = -1;

switch (fd) {
case STDIN_FILENO:
	cap_rights_set(&rights, CAP_READ);
	break;
case STDOUT_FILENO:
	cap_rights_set(&rights, CAP_IOCTL, CAP_WRITE);
	cmd = TIOCGETA; /* required by isatty(3) in printf(3) */
	break;
case STDERR_FILENO:
	cap_rights_set(&rights, CAP_WRITE);
	if (!suppressdata) {
		cap_rights_set(&rights, CAP_IOCTL);
		cmd = TIOCGWINSZ;
	}
	break;
default:
	abort();
}

if (cap_rights_limit(fd, &rights) < 0 && errno != ENOSYS)
	err(1, "unable to limit rights for descriptor %d", fd);
if (cap_ioctls_limit(fd, &cmd, 1) < 0 && errno != ENOSYS)
	err(1, "unable to limit ioctls for descriptor %d", fd);
```

libcapsicum

```
/*
 * The function opens unrestricted communication channel to Casper.
 */
cap_channel_t *cap_init(void);

/*
 * The function creates cap_channel_t based on the given socket.
 * This is useful when a program is executed and have some sockets open.
 */
cap_channel_t *cap_wrap(int sock);

/*
 * The function returns communication socket and frees cap_channel_t.
 */
int     cap_unwrap(cap_channel_t *chan);

/*
 * The function clones the given capability.
 */
cap_channel_t *cap_clone(const cap_channel_t *chan);

/*
 * The function closes the given capability.
 */
void  cap_close(cap_channel_t *chan);
```

libcapsicum

```
/*
 * The function returns socket descriptor associated with the given
 * cap_channel_t for use with select(2)/kqueue(2)/etc.
 */
int	cap_sock(const cap_channel_t *chan);

/*
 * The function limits the given capability. It will destroy nvl on success.
 */
int	cap_limit_set(const cap_channel_t *chan, const nvlist_t *limits);

/*
 * The function returns current limits of the given capability.
 */
nvlist_t *cap_limit_get(const cap_channel_t *chan);
```

libcapsicum

```
/*
 * Function sends nvlist over the given capability.
*/
int      cap_send_nvlist(const cap_channel_t *chan, const nvlist_t *nvl);

/*
 * Function receives nvlist over the given capability.
*/
nvlist_t *cap_recv_nvlist(const cap_channel_t *chan);

/*
 * Function sends the given nvlist, destroys it and receives new nvlist
 * in response over the given capability.
*/
nvlist_t *cap_xfer_nvlist(const cap_channel_t *chan, nvlist_t *nvl);
```

Casper

Casper services

Casper services

```
struct hostent *gethostbyname(const char *name);
struct hostent *gethostbyname2(const char *name, int type);
struct hostent *gethostbyaddr(const void *addr, socklen_t len, int type);

int getaddrinfo(const char *hostname, const char *servname, const struct addrinfo *hints,
        struct addrinfo **res);
int getnameinfo(const struct sockaddr *sa, socklen_t salen, char *host, size_t hostlen,
        char *serv, size_t servlen, int flags);
```

```
struct hostent *cap_gethostbyname(cap_channel_t *chan, const char *name);
struct hostent *cap_gethostbyname2(cap_channel_t *chan, const char *name, int type);
struct hostent *cap_gethostbyaddr(cap_channel_t *chan, const void *addr, socklen_t len, int type);

int cap_getaddrinfo(cap_channel_t *chan, const char *hostname, const char *servname,
        const struct addrinfo *hints, struct addrinfo **res);
int cap_getnameinfo(cap_channel_t *chan, const struct sockaddr *sa, socklen_t salen,
        char *host, size_t hostlen, char *serv, size_t servlen, int flags);
```

tcpdump

```
capchannel_t *capcas, *capdns;

if (nflag) {
        /* -n given? No DNS access required. */
        capcas = NULL;
        capdns = NULL;
} else {
        capcas = cap_init();
        if (capcas == NULL)
                error("unable to contact Casper");
        capdns = cap_service_open(capcas, "system.dns");
        if (capdns == NULL)
                error("unable to open system.dns service");
        /* Limit system.dns to reverse DNS lookups. */
        limits = nvlist_create(0);
        nvlist_add_string(limits, "type", "ADDR");
        nvlist_add_number(limits, "family", (uint64_t)AF_INET);
        nvlist_add_number(limits, "family", (uint64_t)AF_INET6);
        if (cap_limit_set(capdns, limits) < 0)
                error("unable to limit access to system.dns service");
        /* Casper capability is no longer needed. */
        cap_close(capcas);
}
```

Status

auditdistd
dhclient
file, libmagic
hastctl, hastd
kdump
rwho, rwhod
sshd
tcpdump
uniq

Status

system.dns
system.filesystem
system.grp
system.pwd
system.random
system.socket
system.sysctl

Sandboxing-friendliness

strerror(3)
strsignal(3)
localtime(3)
login_cap(3)
getpwent(3)
getgrent(3)
getservent(3)
getprotoent(3)
getrpcent(3)

Questions?

FreeBSD