

**NAME**

**cap\_bind**, **cap\_connect**, **cap\_getaddrinfo**, **cap\_gethostbyaddr**, **cap\_gethostbyname**, **cap\_gethostbyname2**, **cap\_getnameinfo**, **cap\_net\_free**, **cap\_net\_limit**, **cap\_net\_limit\_addr2name**, **cap\_net\_limit\_addr2name\_family**, **cap\_net\_limit\_bind**, **cap\_net\_limit\_connect**, **cap\_net\_limit\_init**, **cap\_net\_limit\_name2addr**, **cap\_net\_limit\_name2addr\_family**, - library for networking in capability mode

**LIBRARY**

library "libcap\_net"

**SYNOPSIS**

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

```
#include <libcasper.h>
```

```
#include <casper/cap_net.h>
```

*int*

```
cap_bind(cap_channel_t *chan, int s, const struct sockaddr *addr, socklen_t addrlen);
```

*int*

```
cap_connect(cap_channel_t *chan, int s, const struct sockaddr *name, socklen_t namelen);
```

*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);
```

*struct hostent* \*

```
cap_gethostbyname(const cap_channel_t *chan, const char *name);
```

*struct hostent* \*

```
cap_gethostbyname2(const cap_channel_t *chan, const char *name, int af);
```

*struct hostent* \*

```
cap_gethostbyaddr(const cap_channel_t *chan, const void *addr, socklen_t len, int af);
```

*cap\_net\_limit\_t* \*

```
cap_net_limit_init(cap_channel_t *chan, uint64_t mode);
```

*int*

**cap\_net\_limit**(*cap\_net\_limit\_t \*limit*);

*void*

**cap\_net\_free**(*cap\_net\_limit\_t \*limit*);

*cap\_net\_limit\_t \**

**cap\_net\_limit\_addr2name\_family**(*cap\_net\_limit\_t \*limit, int \*family, size\_t size*);

*cap\_net\_limit\_t \**

**cap\_net\_limit\_addr2name**(*cap\_net\_limit\_t \*limit, const struct sockaddr \*sa, socklen\_t salen*);

*cap\_net\_limit\_t \**

**cap\_net\_limit\_name2addr\_family**(*cap\_net\_limit\_t \*limit, int \*family, size\_t size*);

*cap\_net\_limit\_t \**

**cap\_net\_limit\_name2addr**(*cap\_net\_limit\_t \*limit, const char \*name, const char \*serv*);

*cap\_net\_limit\_t \**

**cap\_net\_limit\_connect**(*cap\_net\_limit\_t \*limit, const struct sockaddr \*sa, socklen\_t salen*);

*cap\_net\_limit\_t \**

**cap\_net\_limit\_bind**(*cap\_net\_limit\_t \*limit, const struct sockaddr \*sa, socklen\_t salen*);

## DESCRIPTION

The functions **cap\_bind()**, **cap\_connect()**, **cap\_gethostbyname()**, **cap\_gethostbyname2()**, **cap\_gethostbyaddr()** and **cap\_getnameinfo()** provide a set of APIs equivalent to **bind(2)**, **connect(2)**, **gethostbyname(3)**, **gethostbyname2(3)**, **gethostbyaddr(3)** and **getnameinfo(3)** except that a connection to the **system.net** service needs to be provided.

## LIMITS

By default, the **cap\_net** capability provides unrestricted access to the network namespace. Applications typically only require access to a small portion of the network namespace: The **cap\_net\_limit()** function can be used to restrict access to the network. The **cap\_net\_limit\_init()** returns an opaque limit handle used to store a list of capabilities. The restricts the functionality of the service. Modes are encoded using the following flags:

|                  |  |
|------------------|--|
| CAPNET_ADDR2NAME | reverse DNS lookups are allowed with<br><b>cap_getnameinfo</b> |
| CAPNET_NAME2ADDR | name resolution is allowed with                                |

|                             |  |
|-----------------------------|--|
|                             | cap_getaddrinfo  |
| CAPNET_DEPRECATED_ADDR2NAME | reverse DNS lookups are allowed with cap_gethostbyaddr   |
| CAPNET_DEPRECATED_NAME2ADDR | name resolution is allowed with cap_gethostbyname and cap_gethostbyname2   |
| CAPNET_BIND                 | bind syscall is allowed  |
| CAPNET_CONNECT              | connect syscall is allowed   |
| CAPNET_CONNECTDNS           | connect syscall is allowed to the values returned from previous call to the cap_getaddrinfo or cap_gethostbyname |

**cap\_net\_limit\_addr2name\_family()** limits the **cap\_getnameinfo()** and **cap\_gethostbyaddr()** to do reverse DNS lookups to specific family (AF\_INET, AF\_INET6, etc.)

**cap\_net\_limit\_addr2name()** limits the **cap\_getnameinfo()** and **cap\_gethostbyaddr()** to do reverse DNS lookups only on those specific structures.

**cap\_net\_limit\_name2addr\_family()** limits the **cap\_getaddrinfo()**, **cap\_gethostbyname()** and **cap\_gethostbyname2()** to do the name resolution on specific family (AF\_INET, AF\_INET6, etc.)

**cap\_net\_limit\_addr2name()** restricts **cap\_getaddrinfo()**, **cap\_gethostbyname()** and **cap\_gethostbyname2()** to a set of domains.

**cap\_net\_limit\_bind()** limits **cap\_bind()** to bind only on those specific structures.

**cap\_net\_limit\_connect()** limits **cap\_connect()** to connect only on those specific structures. If the CAPNET\_CONNECTDNS is set the limits are extended to the values returned by **cap\_getaddrinfo()**, **cap\_gethostbyname()** and **cap\_gethostbyname2()** In case of the **cap\_getaddrinfo()** the restriction is strict. In case of the **cap\_gethostbyname()** and **cap\_gethostbyname2()** any port will be accepted in the **cap\_connect()** function.

**cap\_net\_limit()** applies a set of sysctl limits to the capability, denying access to sysctl variables not belonging to the set.

Once a set of limits is applied, subsequent calls to **cap\_net\_limit()** will fail unless the new set is a subset of the current set.

The **cap\_net\_limit()** will consume the limits. If the **cap\_net\_limit()** was not called the rights may be freed using **cap\_net\_free()**. Multiple calls to **cap\_net\_limit\_addr2name\_family()**, **cap\_net\_limit\_addr2name()**, **cap\_net\_limit\_name2addr\_family()**, **cap\_net\_limit\_name2addr()**,

**cap\_net\_limit\_connect()**, and **cap\_net\_limit\_bind()** is supported, each call is extending preview capabilities.

## EXAMPLES

The following example first opens a capability to casper and then uses this capability to create the **system.net** casper service and uses it to resolve a host and connect to it.

```
cap_channel_t *capcas, *capnet;
cap_net_limit_t *limit;
int familylimit, error, s;
const char *host = "example.com";
struct addrinfo hints, *res;

/* Open capability to Casper. */
capcas = cap_init();
if (capcas == NULL)
    err(1, "Unable to contact Casper");

/* Cache NLA for gai_strerror. */
caph_cache_catpages();

/* Enter capability mode sandbox. */
if (caph_enter_casper() < 0)
    err(1, "Unable to enter capability mode");

/* Use Casper capability to create capability to the system.net service. */
capnet = cap_service_open(capcas, "system.net");
if (capnet == NULL)
    err(1, "Unable to open system.net service");

/* Close Casper capability. */
cap_close(capcas);

/* Limit system.net to reserve IPv4 addresses, to host example.com . */
limit = cap_net_limit_init(capnet, CAPNET_NAME2ADDR | CAPNET_CONNECTDNS);
if (limit == NULL)
    err(1, "Unable to create limits.");
cap_net_limit_name2addr(limit, host, "80");
familylimit = AF_INET;
cap_net_limit_name2addr_family(limit, &familylimit, 1);
```

```
if (cap_net_limit(limit) < 0)
    err(1, "Unable to apply limits.");

/* Find IP addresses for the given host. */
memset(&hints, 0, sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;

error = cap_getaddrinfo(capnet, host, "80", &hints, &res);
if (error != 0)
    errx(1, "cap_getaddrinfo(): %s: %s", host, gai_strerror(error));

s = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
if (s < 0)
    err(1, "Unable to create socket");

if (cap_connect(capnet, s, res->ai_addr, res->ai_addrlen) < 0)
    err(1, "Unable to connect to host");
```

**SEE ALSO**

bind(2), cap\_enter(2), connect(2), caph\_enter(3), err(3), gethostbyaddr(3), gethostbyname(3), gethostbyname2(3), getnameinfo(3), capsicum(4), nv(9)

**AUTHORS**

Mariusz Zaborski <[oshogbo@FreeBSD.org](mailto:oshogbo@FreeBSD.org)>