

Re: A code from M.Jerzy Buczynski

Source: <http://newsgroups.derkeiler.com/Archive/Comp/comp.bugs.misc/2006-06/msg00004.html>

- *From:* richard_creon@xxxxxxxxxx
 - *Date:* 24 Jun 2006 19:28:28 -0700
-

richard_cr...@xxxxxxxxxx wrote:

richard_cr...@xxxxxxxxxx wrote:

Validate.

```
/*
 * socket.c — socket library functions
 *
 * Copyright 1998 by Eric S. Raymond.
 * For license terms, see the file COPYING in this directory.
 */
```

```
#include "config.h"
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <ctype.h> /* isspace() */
#ifdef HAVE_MEMORY_H
#include <memory.h>
#endif /* HAVE_MEMORY_H */
#include <sys/types.h>
#ifndef HAVE_NET_SOCKET_H
#include <sys/socket.h>
#else
#include <net/socket.h>
#endif
#include <sys/un.h>
#include <netinet/in.h>
#ifdef HAVE_ARPA_INET_H
#include <arpa/inet.h>
#endif
#include <netdb.h>
#ifdef STDC_HEADERS
#include <stdlib.h>
#endif
#ifdef HAVE_UNISTD_H
#include <unistd.h>
```

Re: A code from M.Jerzy Buczynski

```
#endif
#if defined(HAVE_STDARG_H)
#include <stdarg.h>
#else
#include <varargs.h>
#endif
#include "socket.h"
#include "fetchmail.h"
#include "i18n.h"

/* Defines to allow BeOS and Cygwin to play nice... */
#ifdef __BEOS__
static char peeked;
#define fm_close(a) closesocket(a)
#define fm_write(a,b,c) send(a,b,c,0)
#define fm_peek(a,b,c) recv(a,b,c,0)
#define fm_read(a,b,c) recv(a,b,c,0)
#else
#define fm_close(a) close(a)
#define fm_write(a,b,c) write(a,b,c)
#define fm_peek(a,b,c) recv(a,b,c, MSG_PEEK)
#ifdef __CYGWIN__
#define fm_read(a,b,c) cygwin_read(a,b,c)
static ssize_t cygwin_read(int sock, void *buf, size_t count);
#else /* ! __CYGWIN__ */
#define fm_read(a,b,c) read(a,b,c)
#endif /* __CYGWIN__ */
#endif

/* We need to define h_errno only if it is not already */
#ifndef h_errno

#ifdef HAVE_RES_SEARCH
/* some versions of FreeBSD should declare this but don't */
extern int h_errno;
#else
/* pretend we have h_errno to avoid some #ifdef's later */
static int h_errno;
#endif
#endif /* ndef h_errno */

extern int mailserver_socket_temp; /* Socket to close if connect
timeout */

#ifdef NET_SECURITY
#include <net/security.h>
#endif /* NET_SECURITY */

#ifdef HAVE_SOCKETPAIR
char *const *parse_plugin(const char *plugin, const char *host, const
```

Re: A code from M.Jerzy Buczynski

Re: A code from M.Jerzy Buczynski

```
char *service)
{ const char **argvec;
  const char *c, *p;
  char *cp, *plugin_copy;
  unsigned int plugin_copy_len;
  unsigned int plugin_offset = 0, plugin_copy_offset = 0;
  unsigned int i, s = 2 * sizeof(char*), host_count = 0, service_count =
  0;
  unsigned int plugin_len = strlen(plugin);
  unsigned int host_len = strlen(host);
  unsigned int service_len = strlen(service);

  for (c = p = plugin; *c; c++)
  { if (isspace(*c) && !isspace(*p))
    s += sizeof(char*);
    if (*p == '%' && *c == 'h')
      host_count++;
    if (*p == '%' && *c == 'p')
      service_count++;
    p = c;
  }

  plugin_copy_len = plugin_len + host_len * host_count + service_len *
  service_count;
  plugin_copy = malloc(plugin_copy_len + 1);
  if (!plugin_copy)
  {
    report(stderr, GT_("fetchmail: malloc failed\n"));
    return NULL;
  }

  while (plugin_copy_offset < plugin_copy_len)
  { if ((plugin[plugin_offset] == '%') && (plugin[plugin_offset + 1] ==
  'h'))
    { strcpy(plugin_copy + plugin_copy_offset, host);
      plugin_copy_offset += 2;
      plugin_copy_offset += host_len;
    }
    else if ((plugin[plugin_offset] == '%') && (plugin[plugin_offset + 1]
  == 'p'))
    { strcpy(plugin_copy + plugin_copy_offset, service);
      plugin_copy_offset += 2;
      plugin_copy_offset += service_len;
    }
    else
    { plugin_copy[plugin_copy_offset] = plugin[plugin_offset];
      plugin_copy_offset++;
      plugin_copy_offset++;
    }
  }
  plugin_copy[plugin_copy_len] = 0;
```

Re: A code from M.Jerzy Buczynski

```
argvec = malloc(s);
if (!argvec)
{
report(stderr, GT_("fetchmail: malloc failed\n"));
return NULL;
}
memset(argvec, 0, s);
for (c = p = plugin_copy, i = 0; *c; c++)
{ if ((!isspace(*c)) && (c == p ? 1 : isspace(*p))) {
argvec[i] = c;
i++;
}
p = c;
}
for (cp = plugin_copy; *cp; cp++)
{ if (isspace(*cp))
*cp = 0;
}
return (char *const*)argvec;
}

static int handle_plugin(const char *host,
const char *service, const char *plugin)
/* get a socket mediated through a given external command */
{
int fds[2];
char *const *argvec;

/*
* The author of this code, Felix von Leitner
<felix@xxxxxxxxxxxxxxxx>, says:
* he chose socketpair() instead of pipe() because socketpair
creates
* bidirectional sockets while allegedly some pipe()
implementations don't.
*/
if (socketpair(AF_UNIX,SOCK_STREAM,0,fds))
{
report(stderr, GT_("fetchmail: socketpair failed\n"));
return -1;
}
switch (fork()) {
case -1:
/* error */
report(stderr, GT_("fetchmail: fork failed\n"));
return -1;
break;
case 0: /* child */
/* fds[1] is the parent's end; close it for proper EOF
** detection */
```

Re: A code from M.Jerzy Buczynski

Re: A code from M.Jerzy Buczynski

```
(void) close(fds[1]);
if ( ( dup2(fds[0],0) == -1) || (dup2(fds[0],1) == -1) ) {
report(stderr, GT_("dup2 failed\n"));
exit(1);
}
/* fds[0] is now connected to 0 and 1; close it */
(void) close(fds[0]);
if (outlevel >= O_VERBOSE)
report(stderr, GT_("running %s (host %s service %s)\n"), plugin,
host, service);
argvec = parse_plugin(plugin,host,service);
execvp(*argvec, argvec);
report(stderr, GT_("execvp(%s) failed\n"), *argvec);
exit(0);
break;
default: /* parent */
/* NOP */
break;
}
/* fds[0] is the child's end; close it for proper EOF detection */
(void) close(fds[0]);
return fds[1];
}
#endif /* HAVE_SOCKETPAIR */

#ifdef __UNUSED__
#include <sys/time.h>

int SockCheckOpen(int fd)
/* poll given socket; is it selectable? */
{
fd_set r, w, e;
int rt;
struct timeval tv;

for (;;)
{
FD_ZERO(&r); FD_ZERO(&w); FD_ZERO(&e);
FD_SET(fd, &e);

tv.tv_sec = 0; tv.tv_usec = 0;
rt = select(fd+1, &r, &w, &e, &tv);
if (rt == -1 && (errno != EAGAIN && errno != EINTR))
return 0;
if (rt != -1)
return 1;
}
}
#endif /* __UNUSED__ */

int UnixOpen(const char *path)
```

Re: A code from M.Jerzy Buczynski

```
{
int sock = -1;
struct sockaddr_un ad;
memset(&ad, 0, sizeof(ad));
ad.sun_family = AF_UNIX;
strncpy(ad.sun_path, path, sizeof(ad.sun_path)-1);

sock = socket( AF_UNIX, SOCK_STREAM, 0 );
if (sock < 0)
{
h_errno = 0;
return -1;
}

/* Socket opened saved. Usefull if connect timeout
* because it can be closed.
*/
mailserver_socket_temp = sock;

if (connect(sock, (struct sockaddr *) &ad, sizeof(ad)) < 0)
{
int olderr = errno;
fm_close(sock); /* don't use SockClose, no traffic yet */
h_errno = 0;
errno = olderr;
sock = -1;
}

/* No connect timeout, then no need to set mailserver_socket_temp */
mailserver_socket_temp = -1;

return sock;
}

#ifdef INET6_ENABLE
int SockOpen(const char *host, const char *service, const char
*options,
const char *plugin)
{
struct addrinfo *ai, *ai0, req;
int i;
#ifdef NET_SECURITY
void *request = NULL;
int requestlen;
#endif /* NET_SECURITY */

#ifdef HAVE_SOCKETPAIR
if (plugin)
return handle_plugin(host,service,plugin);
#endif /* HAVE_SOCKETPAIR */
memset(&req, 0, sizeof(struct addrinfo));
```

Re: A code from M.Jerzy Buczynski

Re: A code from M.Jerzy Buczynski

```
req.ai_socktype = SOCK_STREAM;

if (getaddrinfo(host, service, &req, &ai0)) {
report(stderr, GT_("fetchmail: getaddrinfo(%s.%s)\n"), host,service);
return -1;
}

#if NET_SECURITY
if (!options)
requestlen = 0;
else
if (net_security_strtorequest((char *)options, &request, &requestlen))
goto ret;

i = inner_connect(ai0, request, requestlen, NULL, NULL,
"fetchmail", NULL);
if (request)
free(request);

ret:
#else /* NET_SECURITY */
#ifdef HAVE_INNER_CONNECT
i = inner_connect(ai0, NULL, 0, NULL, NULL, "fetchmail", NULL);
if (i >= 0)
break;
#else

i = -1;
for (ai = ai0; ai; ai = ai->ai_next) {
i = socket(ai->ai_family, ai->ai_socktype, 0);
if (i < 0)
continue;

/* Socket opened saved. Usefull if connect timeout
* because it can be closed.
*/
mailserver_socket_temp = i;

if (connect(i, (struct sockaddr *) ai->ai_addr, ai->ai_addrlen) < 0) {
fm_close(i);
i = -1;
continue;
}

/* No connect timeout, then no need to set mailserver_socket_temp */
mailserver_socket_temp = -1;

break;
}

#endif
#endif
```

Re: A code from M.Jerzy Buczynski

Re: A code from M.Jerzy Buczynski

```
#endif /* NET_SECURITY */

freeaddrinfo(ai0);

return i;
}
#else /* INET6_ENABLE */
#ifndef HAVE_INET_ATON
#ifndef INADDR_NONE
#ifndef INADDR_BROADCAST
#define INADDR_NONE INADDR_BROADCAST
#else
#define INADDR_NONE -1
#endif
#endif
#endif
#endif /* HAVE_INET_ATON */

int SockOpen(const char *host, int clientPort, const char *options,
const char *plugin)
{
int sock = -1; /* pacify -Wall */
#ifndef HAVE_INET_ATON
unsigned long inaddr;
#endif /* HAVE_INET_ATON */
struct sockaddr_in ad, **pptr;
struct hostent *hp;

#ifdef HAVE_SOCKETPAIR
if (plugin) {
char buf[10];
#ifdef HAVE_SNPRINTF
snprintf(buf, sizeof(buf), /* Yeah, paranoid. So what? :P */
#else
sprintf(buf,
#endif /* HAVE_SNPRINTF */
"%d",clientPort);
return handle_plugin(host,buf,plugin);
}
#endif /* HAVE_SOCKETPAIR */

memset(&ad, 0, sizeof(ad));
ad.sin_family = AF_INET;

/* we'll accept a quad address */
#ifndef HAVE_INET_ATON
inaddr = inet_addr((char*)host);
if (inaddr != INADDR_NONE)
{
memcpy(&ad.sin_addr, &inaddr, sizeof(inaddr));
#else
if (inet_aton(host, &ad.sin_addr))
```

Re: A code from M.Jerzy Buczynski

Re: A code from M.Jerzy Buczynski

```
{
#endif /* HAVE_INET_ATON */
ad.sin_port = htons(clientPort);

sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock < 0)
{
h_errno = 0;
return -1;
}

/* Socket opened saved. Usefull if connect timeout because
* it can be closed
*/
mailserver_socket_temp = sock;

if (connect(sock, (struct sockaddr *) &ad, sizeof(ad)) < 0)
{
int olderr = errno;
fm_close(sock); /* don't use SockClose, no traffic yet */
h_errno = 0;
errno = olderr;
return -1;
}

/* No connect timeout, then no need to set mailserver_socket_temp */
mailserver_socket_temp = -1;

#ifdef HAVE_INET_ATON
}
#else
}
#endif /* HAVE_INET_ATON */
else {
hp = gethostbyname((char*)host);

if (hp == NULL)
{
errno = 0;
return -1;
}
/*
* Add a check to make sure the address has a valid IPv4 or IPv6
* length. This prevents buffer spamming by a broken DNS.
*/
if(hp->h_length != 4 && hp->h_length != 8)
{
h_errno = errno = 0;
report(stderr,
GT_("fetchmail: illegal address length received for host
%s\n"),host);
```

Re: A code from M.Jerzy Buczynski

Re: A code from M.Jerzy Buczynski

```
return -1;
}
/*
 * Try all addresses of a possibly multihomed host until we get
 * a successful connect or until we run out of addresses.
 */
pptr = (struct sockaddr_in **)hp->h_addr_list;
for(; *pptr != NULL; pptr++)
{
sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock < 0)
{
h_errno = 0;
return -1;
}

/* Socket opened saved. Usefull if connect timeout because
 * it can be closed
 */
mailserver_socket_temp = sock;

ad.sin_port = htons(clientPort);
memcpy(&ad.sin_addr, *pptr, sizeof(struct in_addr));
if (connect(sock, (struct sockaddr *) &ad, sizeof(ad)) == 0) {
/* No connect timeout, then no need to set mailserver_socket_temp */
mailserver_socket_temp = -1;
break; /* success */
}
fm_close(sock); /* don't use SockClose, no traffic yet */
memset(&ad, 0, sizeof(ad));
ad.sin_family = AF_INET;
}
if(*pptr == NULL)
{
int olderr = errno;
fm_close(sock); /* don't use SockClose, no traffic yet */
h_errno = 0;
errno = olderr;
return -1;
}
}

return(sock);
}
#endif /* INET6_ENABLE */

#if defined(HAVE_STDARG_H)
int SockPrintf(int sock, const char* format, ...)
{
#else
```

Re: A code from M.Jerzy Buczynski

```
int SockPrintf(sock,format,va_alist)
int sock;
char *format;
va_dcl {
#endif

va_list ap;
char buf[8192];

#ifdef HAVE_STDARG_H
va_start(ap, format) ;
#else
va_start(ap);
#endif
#ifdef HAVE_VSNPRINTF
vsnprintf(buf, sizeof(buf), format, ap);
#else
vsprintf(buf, format, ap);
#endif
va_end(ap);
return SockWrite(sock, buf, strlen(buf));

}

#ifdef SSL_ENABLE
#include "openssl/ssl.h"
#include "openssl/err.h"
#include "openssl/pem.h"
#include "openssl/x509.h"

static SSL_CTX *_ctx = NULL;
static SSL *_ssl_context[FD_SETSIZE];

SSL *SSLGetContext( int );
#endif /* SSL_ENABLE */

int SockWrite(int sock, char *buf, int len)
{
int n, wrlen = 0;
#ifdef SSL_ENABLE
SSL *ssl;
#endif

while (len)
{
#ifdef SSL_ENABLE
if( NULL != ( ssl = SSLGetContext( sock ) ) )
n = SSL_write(ssl, buf, len);
else
#endif /* SSL_ENABLE */
n = fm_write(sock, buf, len);
}
```

Re: A code from M.Jerzy Buczynski

11

Re: A code from M.Jerzy Buczynski

```
if (n <= 0)
return -1;
len -= n;
wrlen += n;
buf += n;
}
return wrlen;
}
```

```
int SockRead(int sock, char *buf, int len)
```

```
{
char *newline, *bp = buf;
int n;
#ifdef SSL_ENABLE
SSL *ssl;
#endif
```

```
if (--len < 1)
return(-1);
#ifdef __BEOS__
if (peeked != 0){
(*bp) = peeked;
bp++;
len--;
peeked = 0;
}
#endif
```

```
do {
```

```
/*
```

```
* The reason for these gymnastics is that we want two things:
```

```
* (1) to read \n-terminated lines,
```

```
* (2) to return the true length of data read, even if the
```

```
* data coming in has embedded NULS.
```

```
*/
```

```
#ifdef SSL_ENABLE
```

```
if( NULL != ( ssl = SSLGetContext( sock ) ) ) {
```

```
/* Hack alert! */
```

```
/* OK... SSL_peek works a little different from MSG_PEEK
```

```
Problem is that SSL_peek can return 0 if there
```

```
is no data currently available. If, on the other
```

```
hand, we loose the socket, we also get a zero, but
```

```
the SSL_read then SEGFAULTS! To deal with this,
```

```
we'll check the error code any time we get a return
```

```
of zero from SSL_peek. If we have an error, we bail.
```

```
If we don't, we read one character in SSL_read and
```

```
loop. This should continue to work even if they
```

```
later change the behavior of SSL_peek
```

```
to "fix" this problem... :-( */
```

```
if ((n = SSL_peek(ssl, bp, len)) < 0) {
```

```
(void)SSL_get_error(ssl, n);
```

```
return(-1);
```

```

}
if( 0 == n ) {
/* SSL_peek says no data... Does he mean no data
or did the connection blow up? If we got an error
then bail! */
if( 0 != ( n = SSL_get_error(ssl, n) ) ) {
return -1;
}
/* We didn't get an error so read at least one
character at this point and loop */
n = 1;
/* Make sure newline start out NULL!
* We don't have a string to pass through
* the strchr at this point yet */
newline = NULL;
} else if ((newline = memchr(bp, '\n', n)) != NULL)
n = newline - bp + 1;
/* Matthias Andree: SSL_read can return 0, in that case
* we must call SSL_get_error to figure if there was
* an error or just a "no data" condition */
if ((n = SSL_read(ssl, bp, n)) <= 0) {
if ((n = SSL_get_error(ssl, n))) {
return(-1);
}
}
/* Check for case where our single character turned out to
* be a newline... (It wasn't going to get caught by
* the strchr above if it came from the hack... ). */
if( NULL == newline && 1 == n && '\n' == *bp ) {
/* Got our newline - this will break
out of the loop now */
newline = bp;
}
}
else
#endif /* SSL_ENABLE */
{

#ifdef __BEOS__
if ((n = fm_read(sock, bp, 1)) <= 0)
#else
if ((n = fm_peek(sock, bp, len)) <= 0)
#endif
return (-1);
if ((newline = memchr(bp, '\n', n)) != NULL)
n = newline - bp + 1;
#ifdef __BEOS__
if ((n = fm_read(sock, bp, n)) == -1)
return(-1);
#endif /* __BEOS__ */
}

```

Re: A code from M.Jerzy Buczynski

```
bp += n;
len -= n;
} while
(!newline && len);
*bp = '\0';
return bp - buf;
}

int SockPeek(int sock)
/* peek at the next socket character without actually reading it */
{
int n;
char ch;
#ifdef SSL_ENABLE
SSL *ssl;
#endif

#ifdef SSL_ENABLE
if( NULL != ( ssl = SSLGetContext( sock ) ) ) {
n = SSL_peek(ssl, &ch, 1);
if (n < 0) {
(void)SSL_get_error(ssl, n);
return -1;
}
if( 0 == n ) {
/* This code really needs to implement a "hold back"
* to simulate a functioning SSL_peek()... sigh...
* Has to be coordinated with the read code above.
* Next on the list todo... */

/* SSL_peek says 0... Does that mean no data
or did the connection blow up? If we got an error
then bail! */
if( 0 != ( n = SSL_get_error(ssl, n) ) ) {
return -1;
}

/* Haven't seen this case actually occur, but...
if the problem in SockRead can occur, this should
be possible... Just not sure what to do here.
This should be a safe "punt" the "peek" but don't
"punt" the "session"... */

return 0; /* Give him a '\0' character */
}
}
else
#endif /* SSL_ENABLE */
n = fm_peek(sock, &ch, 1);
if (n == -1)
return -1;
}
```

Re: A code from M.Jerzy Buczynski

14

```
#ifdef __BEOS__
peeked = ch;
#endif
return(ch);
}

#ifdef SSL_ENABLE

static char *_ssl_server_cname = NULL;
static int _check_fp;
static char *_check_digest;
static char *_server_label;
static int _depth0ck;

SSL *SSLGetContext( int sock )
{
/* If SSLOpen has never initialized – just return NULL */
if( NULL == _ctx )
return NULL;

if( sock < 0 || sock > FD_SETSIZE )
return NULL;
return _ssl_context[sock];
}

int SSL_verify_callback( int ok_return, X509_STORE_CTX *ctx, int strict
)
{
char buf[257];
X509 *x509_cert;
int err, depth;
unsigned char digest[EVP_MAX_MD_SIZE];
char text[EVP_MAX_MD_SIZE * 3 + 1], *tp, *te;
EVP_MD *digest_tp;
unsigned int dsz, i, esz;
X509_NAME *subj, *issuer;

x509_cert = X509_STORE_CTX_get_current_cert(ctx);
err = X509_STORE_CTX_get_error(ctx);
depth = X509_STORE_CTX_get_error_depth(ctx);

subj = X509_get_subject_name(x509_cert);
issuer = X509_get_issuer_name(x509_cert);

if (depth == 0) {
_depth0ck = 1;

if (outlevel == O_VERBOSE) {
if ((i = X509_NAME_get_text_by_NID(issuer, NID_organizationName,
```

Re: A code from M.Jerzy Buczynski

```
buf, sizeof(buf))) != -1) {
report(stdout, GT_("Issuer Organization: %s\n"), buf);
if (i >= sizeof(buf) - 1)
report(stdout, GT_("Warning: Issuer Organization Name too long
(possibly truncated).\n"));
} else
report(stdout, GT_("Unknown Organization\n"));
if ((i = X509_NAME_get_text_by_NID(issuer, NID_commonName, buf,
sizeof(buf))) != -1) {
report(stdout, GT_("Issuer CommonName: %s\n"), buf);
if (i >= sizeof(buf) - 1)
report(stdout, GT_("Warning: Issuer CommonName too long (possibly
truncated).\n"));
} else
report(stdout, GT_("Unknown Issuer CommonName\n"));
}
if ((i = X509_NAME_get_text_by_NID(subj, NID_commonName, buf,
sizeof(buf))) != -1) {
if (outlevel == O_VERBOSE)
report(stdout, GT_("Server CommonName: %s\n"), buf);
if (i >= sizeof(buf) - 1) {
/* Possible truncation. In this case, this is a DNS name, so this
* is really bad. We do not tolerate this even in the non-strict
case. */
report(stderr, GT_("Bad certificate: Subject CommonName too
long!\n"));
return (0);
}
if (_ssl_server_cname != NULL) {
char *p1 = buf;
char *p2 = _ssl_server_cname;
int n;

if (*p1 == '*') {
++p1;
n = strlen(p2) - strlen(p1);
if (n >= 0)
p2 += n;
}
if (0 != strcasecmp(p1, p2)) {
report(stderr,
GT_("Server CommonName mismatch: %s != %s\n"),
buf, _ssl_server_cname);
if (ok_return && strict)
return (0);
}
} else if (ok_return && strict) {
report(stderr, GT_("Server name not set, could not verify
certificate!\n"));
return (0);
}
}
```

Re: A code from M.Jerzy Buczynski

```
} else {
if (outlevel == O_VERBOSE)
report(stdout, GT_("Unknown Server CommonName\n"));
if (ok_return && strict) {
report(stderr, GT_("Server name not specified in certificate!\n"));
return (0);
}
}
/* Print the finger print. Note that on errors, we might print it
more than once
* normally; we kluge around that by using a global variable. */
if (_check_fp) {
_check_fp = 0;
digest_tp = EVP_md5();
if (digest_tp == NULL) {
report(stderr, GT_("EVP_md5() failed!\n"));
return (0);
}
if (!X509_digest(x509_cert, digest_tp, digest, &dsz)) {
report(stderr, GT_("Out of memory!\n"));
return (0);
}
tp = text;
te = text + sizeof(text);
for (i = 0; i < dsz; i++) {
#ifdef HAVE_SNPRINTF
esz = snprintf(tp, te - tp, i > 0 ? ":%02X" : "%02X", digest[i]);
#else
esz = sprintf(tp, i > 0 ? ":%02X" : "%02X", digest[i]);
#endif
if (esz >= te - tp) {
report(stderr, GT_("Digest text buffer too small!\n"));
return (0);
}
tp += esz;
}
if (outlevel > O_NORMAL)
report(stdout, GT_("%s key fingerprint: %s\n"), _server_label,
text);
if (_check_digest != NULL) {
if (strcmp(text, _check_digest) == 0) {
if (outlevel > O_NORMAL)
report(stdout, GT_("%s fingerprints match.\n"), _server_label);
} else {
if (outlevel > O_SILENT)
report(stderr, GT_("%s fingerprints do not match!\n"),
_server_label);
return (0);
}
}
}
}
```

Re: A code from M.Jerzy Buczynski

17

Re: A code from M.Jerzy Buczynski

```
}

if (err != X509_V_OK && (strict || outlevel == O_VERBOSE)) {
report(strict ? stderr : stdout, GT_("Warning: server certificate
verification: %s\n"), X509_verify_cert_error_string(err));
/* We gave the error code, but maybe we can add some more details for
debugging */
switch (err) {
case X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT:
X509_NAME_oneline(issuer, buf, sizeof(buf));
buf[sizeof(buf) - 1] = '\0';
report(stdout, GT_("unknown issuer (first %d characters): %s\n"),
sizeof(buf), buf);
break;
}
}
if (!strict)
ok_return = 1;
return (ok_return);
}

int SSL_nock_verify_callback( int ok_return, X509_STORE_CTX *ctx )
{
return SSL_verify_callback(ok_return, ctx, 0);
}

int SSL_ck_verify_callback( int ok_return, X509_STORE_CTX *ctx )
{
return SSL_verify_callback(ok_return, ctx, 1);
}

/* performs initial SSL handshake over the connected socket
* uses SSL *ssl global variable, which is currently defined
* in this file
*/
int SSLOpen(int sock, char *mycert, char *mykey, char *myproto, int
certck, char *certpath,
char *fingerprint, char *servername, char *label)
{
SSL *ssl;

SSL_load_error_strings();
SSLeyay_add_ssl_algorithms();

if( sock < 0 || sock > FD_SETSIZE ) {
report(stderr, GT_("File descriptor out of range for SSL" ));
return( -1 );
}

if( !_ctx ) {
/* Be picky and make sure the memory is cleared */
```

Re: A code from M.Jerzy Buczynski

```
memset( _ssl_context, 0, sizeof( _ssl_context ) );
if(myproto) {
if(!strcmp("ssl2",myproto)) {
_ctx = SSL_CTX_new(SSLv2_client_method());
} else if(!strcmp("ssl3",myproto)) {
_ctx = SSL_CTX_new(SSLv3_client_method());
} else if(!strcmp("tls1",myproto)) {
_ctx = SSL_CTX_new(TLSv1_client_method());
} else {
fprintf(stderr,GT_("Invalid SSL protocol '%s' specified, using
default (SSLv23).\n"), myproto);
myproto = NULL;
}
}
if(!myproto) {
_ctx = SSL_CTX_new(SSLv23_client_method());
}
if(_ctx == NULL) {
ERR_print_errors_fp(stderr);
return(-1);
}
}

if (certck) {
SSL_CTX_set_verify(_ctx, SSL_VERIFY_PEER, SSL_ck_verify_callback);
if (certpath)
SSL_CTX_load_verify_locations(_ctx, NULL, certpath);
} else {
/* In this case, we do not fail if verification fails. However,
* we provide the callback for output and possible fingerprint
checks. */
SSL_CTX_set_verify(_ctx, SSL_VERIFY_PEER,
SSL_nock_verify_callback);
}

_ssl_context[sock] = SSL_new(_ctx);

if(_ssl_context[sock] == NULL) {
ERR_print_errors_fp(stderr);
return(-1);
}

/* This static is for the verify callback */
_ssl_server_cname = servercname;
_server_label = label;
_check_fp = 1;
_check_digest = fingerprint;
_depth0ck = 0;

if( mycert || mykey ) {
```

Re: A code from M.Jerzy Buczynski

```
/* Ok... He has a certificate file defined, so lets declare it. If
 * he does NOT have a separate certificate and private key file then
 * assume that it's a combined key and certificate file.
 */
if( !mykey )
mykey = mycert;
if( !mycert )
mycert = mykey;
SSL_use_certificate_file(_ssl_context[sock], mycert,
SSL_FILETYPE_PEM);
SSL_use_RSAPrivateKey_file(_ssl_context[sock], mykey,
SSL_FILETYPE_PEM);
}

SSL_set_fd(_ssl_context[sock], sock);

if(SSL_connect(_ssl_context[sock]) == -1) {
ERR_print_errors_fp(stderr);
return(-1);
}

/* Paranoia: was the callback not called as we expected? */
if ((fingerprint != NULL || certck) && !_depth0ck) {
report(stderr, GT_("Certificate/fingerprint verification was somehow
skipped!\n"));
}

if( NULL != ( ssl = SSLGetContext( sock ) ) ) {
/* Clean up the SSL stack */
SSL_free( _ssl_context[sock] );
_ssl_context[sock] = NULL;
}
return(-1);
}

return(0);
}
#endif

int SocketClose(int sock)
/* close a socket gracefully */
{
#ifdef SSL_ENABLE
SSL *ssl;

if( NULL != ( ssl = SSLGetContext( sock ) ) ) {
/* Clean up the SSL stack */
SSL_free( _ssl_context[sock] );
_ssl_context[sock] = NULL;
}
#endif
}
#endif
```

Re: A code from M.Jerzy Buczynski

```
#ifndef __UNUSED__
/*
 * This hangs in RedHat 6.2 after fetchmail runs for a while a
 * FIN_WAIT2 comes up in netstat and fetchmail never returns from
 * the recv system call. (Reported from jtnews
 * <jtnews@xxxxxxxxxxxxxxxxxx>, Wed, 24 May 2000 21:26:02.)
 *
 * Half-close the connection first so the other end gets notified.
 *
 * This stops sends but allows receives (effectively, it sends a
 * TCP <FIN>). */
if (shutdown(sock, 1) == 0) {
char ch;
/* If there is any data still waiting in the queue, discard it.
 * Call recv() until either it returns 0 (meaning we received a FIN)
 * or any error occurs. This makes sure all data sent by the other
 * side is acknowledged at the TCP level.
 */
if (fm_peek(sock, &ch, 1) > 0)
while (fm_read(sock, &ch, 1) > 0)
continue;
}
#endif /* __UNUSED__ */

/* if there's an error closing at this point, not much we can do */
return(fm_close(sock)); /* this is guarded */
}

#ifdef __CYGWIN__
/*
 * Workaround Microsoft Winsock recv/WSARecv(..., MSG_PEEK) bug.
 * See http://sources.redhat.com/ml/cygwin/2001-08/msg00628.html
 * for more details.
 */
static ssize_t cygwin_read(int sock, void *buf, size_t count)
{
char *bp = buf;
int n = 0;

if ((n = read(sock, bp, count)) == -1)
return(-1);

if (n != count) {
int n2 = 0;
if (outlevel >= O_VERBOSE)
report(stdout, GT_("Cygwin socket read retry\n"));
n2 = read(sock, bp + n, count - n);
if (n2 == -1 || n + n2 != count) {
report(stderr, GT_("Cygwin socket read retry failed!\n"));
return(-1);
}
}
}

```

Re: A code from M.Jerzy Buczynski

```
}  
}  
#endif /* __CYGWIN__ */  
  
#ifdef MAIN  
/*  
 * Use the chargen service to test input buffering directly.  
 * You may have to uncomment the `chargen' service description in your  
 * inetd.conf (and then SIGHUP inetd) for this to work. */  
main()  
{  
int sock = SockOpen("localhost", 19, NULL);  
char buf[80];  
  
while (SockRead(sock, buf, sizeof(buf)-1))  
SockWrite(1, buf, strlen(buf));  
SockClose(sock);  
}  
#endif /* MAIN */  
  
/* socket.c ends here */
```