dropbear / src / cli-main.c

mkj Execute multihop commands directly, no shell e5a0ef2 · yesterday

176 lines (150 l…

Code Blame Raw

```
/*
 * Dropbear - a SSH2 server
 * SSH client implementation
 *
 * Copyright (c) 2002,2003 Matt Johnston
 * Copyright (c) 2004 by Mihnea Stoenescu
 * All rights reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE. */

#include "includes.h"
#include "dbutil.h"
#include "runopts.h"
#include "session.h"
#include "dbrandom.h"
#include "crypto_desc.h"
#include "fuzz.h"

#if DROPBEAR_CLI_PROXYCMD
```

```
37	static void cli_proxy_cmd(int *sock_in, int *sock_out, pid_t *pid_out);
38	static void kill_proxy_sighandler(int signo);
39	#endif
40
41	#if defined(DBMULTI_dbclient) || !DROPBEAR_MULTI
42	#if defined(DBMULTI_dbclient) && DROPBEAR_MULTI
43	int cli_main(int argc, char ** argv) {
44	#else
45	int main(int argc, char ** argv) {
46	#endif
47
48		int sock_in, sock_out;
49		struct dropbear_progress_connection *progress = NULL;
50		pid_t proxy_cmd_pid = 0;
51
52		_dropbear_exit = cli_dropbear_exit;
53		_dropbear_log = cli_dropbear_log;
54
55		disallow_core();
56
57		seedrandom();
58		crypto_init();
59
60		cli_getopts(argc, argv);
61
62	#ifndef DISABLE_SYSLOG
63		if (opts.usingsyslog) {
64			startsyslog("dbclient");
65		}
66	#endif
67
68		if (cli_opts.bind_address) {
69			DEBUG1(("connect to: user=%s host=%s/%s bind_address=%s:%s", cli_opts.username,
70				cli_opts.remotehost, cli_opts.remoteport, cli_opts.bind_address, cli_opt
71		} else {
72			DEBUG1(("connect to: user=%s host=%s/%s",cli_opts.username,cli_opts.remotehost,c
73		}
74
75		if (signal(SIGPIPE, SIG_IGN) == SIG_ERR) {
76			dropbear_exit("signal() error");
77		}
78
79	#if DROPBEAR_CLI_PROXYCMD
80		if (cli_opts.proxycmd || cli_opts.proxyexec) {
81			cli_proxy_cmd(&sock_in, &sock_out, &proxy_cmd_pid);
82			if (signal(SIGINT, kill_proxy_sighandler) == SIG_ERR ||
83				signal(SIGTERM, kill_proxy_sighandler) == SIG_ERR ||
84				signal(SIGHUP, kill_proxy_sighandler) == SIG_ERR) {
85				dropbear_exit("signal() error");
86			}
87		} else
88	#endif
89		{
```

```
90                  progress = connect_remote(cli_opts.remotehost, cli_opts.remoteport,
91                          cli_connected, &ses, cli_opts.bind_address, cli_opts.bind_port,
92                          DROPBEAR_PRIO_LOWDELAY);
93                  sock_in = sock_out = -1;
94          }
95
96          cli_session(sock_in, sock_out, progress, proxy_cmd_pid);
97
98          /* not reached */
99          return -1;
100 }
101 #endif /* DBMULTI stuff */
102
103 #if DROPBEAR_CLI_PROXYCMD
104 static void shell_proxy_cmd(const void *user_data_cmd) {
105         const char *cmd = user_data_cmd;
106         char *usershell;
107
108         usershell = m_strdup(get_user_shell());
109         run_shell_command(cmd, ses.maxfd, usershell);
110         dropbear_exit("Failed to run '%s'\n", cmd);
111 }
112
113 static void exec_proxy_cmd(const void *unused) {
114         (void)unused;
115         run_command(cli_opts.proxyexec[0], cli_opts.proxyexec, ses.maxfd);
116         dropbear_exit("Failed to run '%s'\n", cli_opts.proxyexec[0]);
117 }
118
119 static void cli_proxy_cmd(int *sock_in, int *sock_out, pid_t *pid_out) {
120         char * cmd_arg = NULL;
121         void (*exec_fn)(const void *user_data) = NULL;
122         int ret;
123
124         /* exactly one of cli_opts.proxycmd or cli_opts.proxyexec should be set */
125
126         /* File descriptor "-j &3" */
127         if (cli_opts.proxycmd && *cli_opts.proxycmd == '&') {
128                 char *p = cli_opts.proxycmd + 1;
129                 int sock = strtoul(p, &p, 10);
130                 /* must be a single number, and not stdin/stdout/stderr */
131                 if (sock > 2 && sock < 1024 && *p == '\0') {
132                         *sock_in = sock;
133                         *sock_out = sock;
134                         goto cleanup;
135                 }
136         }
137
138         if (cli_opts.proxycmd) {
139                 /* Normal proxycommand */
140                 size_t shell_cmdlen;
141                 /* So that spawn_command knows which shell to run */
142                 fill_passwd(cli_opts.own_user);
```

```
143
144                     shell_cmdlen = strlen(cli_opts.proxycmd) + 6; /* "exec " + command + '\0' */
145                     cmd_arg = m_malloc(shell_cmdlen);
146                     snprintf(cmd_arg, shell_cmdlen, "exec %s", cli_opts.proxycmd);
147                     exec_fn = shell_proxy_cmd;
148             } else {
149                     /* No shell */
150                     exec_fn = exec_proxy_cmd;
151             }
152
153             ret = spawn_command(exec_fn, cmd_arg, sock_out, sock_in, NULL, pid_out);
154             if (ret == DROPBEAR_FAILURE) {
155                     dropbear_exit("Failed running proxy command");
156                     *sock_in = *sock_out = -1;
157             }
158
159     cleanup:
160             m_free(cli_opts.proxycmd);
161             m_free(cmd_arg);
162             if (cli_opts.proxyexec) {
163                     char **a = NULL;
164                     for (a = cli_opts.proxyexec; *a; a++) {
165                             m_free_direct(*a);
166                     }
167                     m_free(cli_opts.proxyexec);
168             }
169     }
170
171     static void kill_proxy_sighandler(int UNUSED(signo)) {
172             kill_proxy_command();
173             _exit(1);
174     }
175
176     #endif /* DROPBEAR_CLI_PROXYCMD */
```