NAME

FascistCheck - check a potential password for guessability

LIBRARY

Cracklib (libcrack, -lcrack)

SYNOPSIS

#include <packer.h>

char *FascistCheck(char *pw, char *dictpath)

DESCRIPTION

CrackLib is a library containing a C function which may be used in a **passwd**(1)-like program.

The idea is simple: try to prevent users from choosing passwords that could be guessed by **Crack** by filtering them out, at source.

FascistCheck() takes two arguments:

pw a string containing the user's chosen "potential password"

dictpath the full path name of the **CrackLib** dictionary, without the suffix

CrackLib is an offshoot of the the version 5 **Crack** software, and contains a considerable number of ideas nicked from the new software.

CrackLib makes literally hundreds of tests to determine whether you've chosen a bad password.

- It tries to generate words from your username and gecos entry to try to match them against what you've chosen.
- It checks for simplistic patterns.
- It then tries to reverse-engineer your password into a dictionary word, and searches for it in your dictionary.

After all that, it's *probably* a safe(-ish) password.

RETURN VALUE

FascistCheck() returns the NULL pointer for a good password or a pointer to a diagnostic string if it is a bad password.

BUGS

It can’t catch everything. Just most things.

It calls getpwuid(getuid()) to look up the user, which *may* affect poorly written programs.

Using more than one dictionary file, *e.g.*:

```
char *msg;

if (msg = FascistCheck(pw, "onepath") ||
    msg = FascistCheck(pw, "anotherpath")) {
        printf("Bad Password: because %s\n", msg);
}
```

works, but it’s a kludge. **Avoid it if possible.** Using just the one dictionary is more efficient, anyway.

PWOpen() routines should cope with having more than one dictionary open at a time.

SEE ALSO

passwd(1), **getpwuid**(3),