Sign in

openssh / **openssh-portable** Public

Notifications | Fork 1.8k | Star 3.1k

Code | Pull requests 110 | Actions | Security | Insights

openssh-portable / **ssherr.c**

djmdjm upstream: improve the error message for u2f enrollment errors by … 59d01f1 · 5 years ago

151 lines (149 loc) · 5.17 KB

Code | Blame | Raw

```c
/*	$OpenBSD: ssherr.c,v 1.10 2020/01/25 23:13:09 djm Exp $ */
/*
 * Copyright (c) 2011 Damien Miller
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */

#include <errno.h>
#include <string.h>
#include "ssherr.h"

const char *
ssh_err(int n)
{
	switch (n) {
	case SSH_ERR_SUCCESS:
```

```
                return "success";
        case SSH_ERR_INTERNAL_ERROR:
                return "unexpected internal error";
        case SSH_ERR_ALLOC_FAIL:
                return "memory allocation failed";
        case SSH_ERR_MESSAGE_INCOMPLETE:
                return "incomplete message";
        case SSH_ERR_INVALID_FORMAT:
                return "invalid format";
        case SSH_ERR_BIGNUM_IS_NEGATIVE:
                return "bignum is negative";
        case SSH_ERR_STRING_TOO_LARGE:
                return "string is too large";
        case SSH_ERR_BIGNUM_TOO_LARGE:
                return "bignum is too large";
        case SSH_ERR_ECPOINT_TOO_LARGE:
                return "elliptic curve point is too large";
        case SSH_ERR_NO_BUFFER_SPACE:
                return "insufficient buffer space";
        case SSH_ERR_INVALID_ARGUMENT:
                return "invalid argument";
        case SSH_ERR_KEY_BITS_MISMATCH:
                return "key bits do not match";
        case SSH_ERR_EC_CURVE_INVALID:
                return "invalid elliptic curve";
        case SSH_ERR_KEY_TYPE_MISMATCH:
                return "key type does not match";
        case SSH_ERR_KEY_TYPE_UNKNOWN:
                return "unknown or unsupported key type";
        case SSH_ERR_EC_CURVE_MISMATCH:
                return "elliptic curve does not match";
        case SSH_ERR_EXPECTED_CERT:
                return "plain key provided where certificate required";
        case SSH_ERR_KEY_LACKS_CERTBLOB:
                return "key lacks certificate data";
        case SSH_ERR_KEY_CERT_UNKNOWN_TYPE:
                return "unknown/unsupported certificate type";
        case SSH_ERR_KEY_CERT_INVALID_SIGN_KEY:
                return "invalid certificate signing key";
        case SSH_ERR_KEY_INVALID_EC_VALUE:
                return "invalid elliptic curve value";
        case SSH_ERR_SIGNATURE_INVALID:
                return "incorrect signature";
        case SSH_ERR_LIBCRYPTO_ERROR:
                return "error in libcrypto";  /* XXX fetch and return */
        case SSH_ERR_UNEXPECTED_TRAILING_DATA:
                return "unexpected bytes remain after decoding";
        case SSH_ERR_SYSTEM_ERROR:
                return strerror(errno);
        case SSH_ERR_KEY_CERT_INVALID:
                return "invalid certificate";
        case SSH_ERR_AGENT_COMMUNICATION:
                return "communication with agent failed";
        case SSH_ERR_AGENT_FAILURE:
                return "agent refused operation";
        case SSH_ERR_DH_GEX_OUT_OF_RANGE:
                return "DH GEX group out of range";
        case SSH_ERR_DISCONNECTED:
                return "disconnected";
        case SSH_ERR_MAC_INVALID:
                return "message authentication code incorrect";
        case SSH_ERR_NO_CIPHER_ALG_MATCH:
                return "no matching cipher found";
        case SSH_ERR_NO_MAC_ALG_MATCH:
                return "no matching MAC found";
        case SSH_ERR_NO_COMPRESS_ALG_MATCH:
                return "no matching compression method found";
        case SSH_ERR_NO_KEX_ALG_MATCH:
                return "no matching key exchange method found";
        case SSH_ERR_NO_HOSTKEY_ALG_MATCH:
                return "no matching host key type found";
        case SSH_ERR_PROTOCOL_MISMATCH:
                return "protocol version mismatch";
        case SSH_ERR_NO_PROTOCOL_VERSION:
                return "could not read protocol version";
        case SSH_ERR_NO_HOSTKEY_LOADED:
                return "could not load host key";
        case SSH_ERR_NEED_REKEY:
                return "rekeying not supported by peer";
        case SSH_ERR_PASSPHRASE_TOO_SHORT:
                return "passphrase is too short (minimum five characters)";
        case SSH_ERR_FILE_CHANGED:
                return "file changed while reading";
        case SSH_ERR_KEY_UNKNOWN_CIPHER:
                return "key encrypted using unsupported cipher";
        case SSH_ERR_KEY_WRONG_PASSPHRASE:
                return "incorrect passphrase supplied to decrypt private key";
        case SSH_ERR_KEY_BAD_PERMISSIONS:
                return "bad permissions";
        case SSH_ERR_KEY_CERT_MISMATCH:
                return "certificate does not match key";
        case SSH_ERR_KEY_NOT_FOUND:
```

```c
	case SSH_ERR_KEY_NOT_FOUND:
		return "key not found";
	case SSH_ERR_AGENT_NOT_PRESENT:
		return "agent not present";
	case SSH_ERR_AGENT_NO_IDENTITIES:
		return "agent contains no identities";
	case SSH_ERR_BUFFER_READ_ONLY:
		return "internal error: buffer is read-only";
	case SSH_ERR_KRL_BAD_MAGIC:
		return "KRL file has invalid magic number";
	case SSH_ERR_KEY_REVOKED:
		return "Key is revoked";
	case SSH_ERR_CONN_CLOSED:
		return "Connection closed";
	case SSH_ERR_CONN_TIMEOUT:
		return "Connection timed out";
	case SSH_ERR_CONN_CORRUPT:
		return "Connection corrupted";
	case SSH_ERR_PROTOCOL_ERROR:
		return "Protocol error";
	case SSH_ERR_KEY_LENGTH:
		return "Invalid key length";
	case SSH_ERR_NUMBER_TOO_LARGE:
		return "number is too large";
	case SSH_ERR_SIGN_ALG_UNSUPPORTED:
		return "signature algorithm not supported";
	case SSH_ERR_FEATURE_UNSUPPORTED:
		return "requested feature not supported";
	case SSH_ERR_DEVICE_NOT_FOUND:
		return "device not found";
	default:
		return "unknown error";
	}
}
```