**auth2.c**

Some code for authentication is changed (and / or deleted) and replaced by authentication code for authentication using GSSAPI.

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff auth2.c ../../moonshot/openssh/auth2.c

1c1

< /\* $OpenBSD: auth2.c,v 1.123 2011/03/10 02:52:57 djm Exp $ \*/

---

> /\* $OpenBSD: auth2.c,v 1.122 2010/08/31 09:58:37 djm Exp $ \*/

71a72

> extern Authmethod method\_gsskeyex;

81a83

> &method\_gsskeyex,

233,234c235,258

< if (authctxt->attempt++ == 0) {

< /\* setup auth context \*/

---

> /\* If first time or username changed or empty username,

>   setup/reset authentication context. \*/

> if ((authctxt->attempt++ == 0) ||

>    (strcmp(user, authctxt->user) != 0) ||

>    (strcmp(user, "") == 0)) {

> if (authctxt->user) {

>    xfree(authctxt->user);

>    authctxt->user = NULL;

> }

> authctxt->valid = 0;

>         authctxt->user = xstrdup(user);

>         if (strcmp(service, "ssh-connection") != 0) {

>             packet\_disconnect("Unsupported service %s", service);

>         }

> #ifdef GSSAPI

> /\* If we're going to set the username based on the

>   GSSAPI context later, then wait until then to

>   verify it. Just put in placeholders for now. \*/

> if ((strcmp(user, "") == 0) &&

>    (strcmp(method, "gssapi-with-mic") == 0 ||

>     strcmp(method, "gssapi-keyex") == 0)) {

> authctxt->pw = fakepw();

> } else {

> #endif

236,237c260

< authctxt->user = xstrdup(user);

< if (authctxt->pw && strcmp(service, "ssh-connection")==0) {

---

> if (authctxt->pw) {

250a274,276

> #ifdef GSSAPI

> } /\* endif for setting username based on GSSAPI context \*/

> #endif

253,259c279,287

< authctxt->service = xstrdup(service);

< authctxt->style = style ? xstrdup(style) : NULL;

< if (use\_privsep)

< mm\_inform\_authserv(service, style);

< userauth\_banner();

< } else if (strcmp(user, authctxt->user) != 0 ||

<    strcmp(service, authctxt->service) != 0) {

---

>                 if (authctxt->attempt == 1) {

> authctxt->service = xstrdup(service);

> authctxt->style = style ? xstrdup(style) : NULL;

> if (use\_privsep)

> mm\_inform\_authserv(service, style);

> userauth\_banner();

> }

> }

> if (strcmp(service, authctxt->service) != 0) {

**authfile.c**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff authfile.c ../../moonshot/openssh/authfile.c

1c1

< /\* $OpenBSD: authfile.c,v 1.92 2011/06/14 22:49:18 markus Exp $ \*/

---

> /\* $OpenBSD: authfile.c,v 1.87 2010/11/29 18:57:04 markus Exp $ \*/

72,73d71

< #define MAX\_KEY\_FILE\_SIZE (1024 \* 1024)

<

282d279

< Buffer copy;

299,301c296

< buffer\_init(&copy);

< buffer\_append(&copy, buffer\_ptr(blob), buffer\_len(blob));

< buffer\_consume(&copy, sizeof(authfile\_id\_string));

---

> buffer\_consume(blob, sizeof(authfile\_id\_string));

304,305c299,300

< (void) buffer\_get\_char(&copy); /\* cipher type \*/

< (void) buffer\_get\_int(&copy); /\* reserved \*/

---

> (void) buffer\_get\_char(blob); /\* cipher type \*/

> (void) buffer\_get\_int(blob); /\* reserved \*/

308c303

< (void) buffer\_get\_int(&copy);

---

> (void) buffer\_get\_int(blob);

310,311c305,306

< buffer\_get\_bignum(&copy, pub->rsa->n);

< buffer\_get\_bignum(&copy, pub->rsa->e);

---

> buffer\_get\_bignum(blob, pub->rsa->n);

> buffer\_get\_bignum(blob, pub->rsa->e);

313c308

< \*commentp = buffer\_get\_string(&copy, NULL);

---

> \*commentp = buffer\_get\_string(blob, NULL);

315c310

< buffer\_free(&copy);

---

> buffer\_clear(blob);

320,321c315,316

< /\* Load a key from a fd into a buffer \*/

< int

---

> /\* Load the contents of a key file into a buffer \*/

> static int

324d318

< u\_char buf[1024];

325a320

> u\_char \*cp;

332a328

> close(fd);

335,337c331

< if ((st.st\_mode & (S\_IFSOCK|S\_IFCHR|S\_IFIFO)) == 0 &&

<    st.st\_size > MAX\_KEY\_FILE\_SIZE) {

<  toobig:

---

> if (st.st\_size > 1\*1024\*1024) {

340a335

> close(fd);

342a338,339

> len = (size\_t)st.st\_size; /\* truncated \*/

>

344,367c341,347

< for (;;) {

< if ((len = atomicio(read, fd, buf, sizeof(buf))) == 0) {

< if (errno == EPIPE)

< break;

< debug("%s: read from key file %.200s%sfailed: %.100s",

<    \_\_func\_\_, filename == NULL ? "" : filename,

<    filename == NULL ? "" : " ", strerror(errno));

< buffer\_clear(blob);

< bzero(buf, sizeof(buf));

< return 0;

< }

< buffer\_append(blob, buf, len);

< if (buffer\_len(blob) > MAX\_KEY\_FILE\_SIZE) {

< buffer\_clear(blob);

< bzero(buf, sizeof(buf));

< goto toobig;

< }

< }

< bzero(buf, sizeof(buf));

< if ((st.st\_mode & (S\_IFSOCK|S\_IFCHR|S\_IFIFO)) == 0 &&

<    st.st\_size != buffer\_len(blob)) {

< debug("%s: key file %.200s%schanged size while reading",

<    \_\_func\_\_, filename == NULL ? "" : filename,

<    filename == NULL ? "" : " ");

---

> cp = buffer\_append\_space(blob, len);

>

> if (atomicio(read, fd, cp, len) != len) {

> debug("%s: read from key file %.200s%sfailed: %.100s", \_\_func\_\_,

>    filename == NULL ? "" : filename,

>    filename == NULL ? "" : " ",

>    strerror(errno));

368a349

> close(fd);

371d351

<

426d405

< Buffer copy;

443,445c422

< buffer\_init(&copy);

< buffer\_append(&copy, buffer\_ptr(blob), buffer\_len(blob));

< buffer\_consume(&copy, sizeof(authfile\_id\_string));

---

> buffer\_consume(blob, sizeof(authfile\_id\_string));

448,449c425,426

< cipher\_type = buffer\_get\_char(&copy);

< (void) buffer\_get\_int(&copy); /\* Reserved data. \*/

---

> cipher\_type = buffer\_get\_char(blob);

> (void) buffer\_get\_int(blob); /\* Reserved data. \*/

452c429

< (void) buffer\_get\_int(&copy);

---

> (void) buffer\_get\_int(blob);

455,456c432,433

< buffer\_get\_bignum(&copy, prv->rsa->n);

< buffer\_get\_bignum(&copy, prv->rsa->e);

---

> buffer\_get\_bignum(blob, prv->rsa->n);

> buffer\_get\_bignum(blob, prv->rsa->e);

458c435

< \*commentp = buffer\_get\_string(&copy, NULL);

---

> \*commentp = buffer\_get\_string(blob, NULL);

460c437

< (void)buffer\_get\_string\_ptr(&copy, NULL);

---

> (void)buffer\_get\_string\_ptr(blob, NULL);

466d442

< buffer\_free(&copy);

471c447

< cp = buffer\_append\_space(&decrypted, buffer\_len(&copy));

---

> cp = buffer\_append\_space(&decrypted, buffer\_len(blob));

477c453

<    buffer\_ptr(&copy), buffer\_len(&copy));

---

>    buffer\_ptr(blob), buffer\_len(blob));

480c456

< buffer\_free(&copy);

---

> buffer\_clear(blob);

633c609

< error("It is required that your private key files are NOT accessible by others.");

---

> error("It is recommended that your private key files are NOT accessible by others.");

653d628

< error("%s: cannot parse key type %d", \_\_func\_\_, type);

698,720d672

< key\_parse\_private(Buffer \*buffer, const char \*filename,

<     const char \*passphrase, char \*\*commentp)

< {

< Key \*pub, \*prv;

<

< /\* it's a SSH v1 key if the public key part is readable \*/

< pub = key\_parse\_public\_rsa1(buffer, commentp);

< if (pub == NULL) {

< prv = key\_parse\_private\_type(buffer, KEY\_UNSPEC,

<    passphrase, NULL);

< /\* use the filename as a comment for PEM \*/

< if (commentp && prv)

< \*commentp = xstrdup(filename);

< } else {

< key\_free(pub);

< /\* key\_parse\_public\_rsa1() has already loaded the comment \*/

< prv = key\_parse\_private\_type(buffer, KEY\_RSA1, passphrase,

<    NULL);

< }

< return prv;

< }

<

< Key \*

724,725c676,677

< Key \*prv;

< Buffer buffer;

---

> Key \*pub, \*prv;

> Buffer buffer, pubcopy;

748c700,716

< prv = key\_parse\_private(&buffer, filename, passphrase, commentp);

---

> buffer\_init(&pubcopy);

> buffer\_append(&pubcopy, buffer\_ptr(&buffer), buffer\_len(&buffer));

> /\* it's a SSH v1 key if the public key part is readable \*/

> pub = key\_parse\_public\_rsa1(&pubcopy, commentp);

> buffer\_free(&pubcopy);

> if (pub == NULL) {

> prv = key\_parse\_private\_type(&buffer, KEY\_UNSPEC,

>    passphrase, NULL);

> /\* use the filename as a comment for PEM \*/

> if (commentp && prv)

> \*commentp = xstrdup(filename);

> } else {

> key\_free(pub);

> /\* key\_parse\_public\_rsa1() has already loaded the comment \*/

> prv = key\_parse\_private\_type(&buffer, KEY\_RSA1, passphrase,

>    NULL);

> }

772,774d739

< /\* Abort loading if this looks like a private key \*/

< if (strncmp(cp, "-----BEGIN", 10) == 0)

< break;

780,784c745,746

< cp[strcspn(cp, "\r\n")] = '\0';

< if (commentp) {

< \*commentp = xstrdup(\*cp ?

<    cp : filename);

< }

---

> if (commentp)

> \*commentp=xstrdup(filename);

**clientloop.c**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff clientloop.c ../../moonshot/openssh/clientloop.c

1c1

< /\* $OpenBSD: clientloop.c,v 1.236 2011/06/22 22:08:42 djm Exp $ \*/

---

> /\* $OpenBSD: clientloop.c,v 1.231 2011/01/16 12:05:59 djm Exp $ \*/

113a114,117

> #ifdef GSSAPI

> #include "ssh-gss.h"

> #endif

>

132a137,139

> /\* Force TTY allocation \*/

> extern int force\_tty\_flag;

>

179,180c186

< int id;

< enum confirm\_action action;

---

> int id, do\_close;

266c272

<    || options.control\_persist\_timeout == 0) {

---

>    || options.control\_persist\_timeout == 0)

269c275

< } else if (channel\_still\_open()) {

---

> else if (channel\_still\_open()) {

663c669

< leave\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

684c690

< enter\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> enter\_raw\_mode(force\_tty\_flag);

743,751d748

< /\*

< \* If a TTY was explicitly requested, then a failure to allocate

< \* one is fatal.

< \*/

< if (cr->action == CONFIRM\_TTY &&

<    (options.request\_tty == REQUEST\_TTY\_FORCE ||

<    options.request\_tty == REQUEST\_TTY\_YES))

< cr->action = CONFIRM\_CLOSE;

<

769c766

< if (cr->action == CONFIRM\_CLOSE && c->self == session\_ident)

---

> if (cr->do\_close && c->self == session\_ident)

771,778c768,771

< /\*

< \* If error occurred on mux client, append to

< \* their stderr.

< \*/

< if (tochan) {

< buffer\_append(&c->extended, errmsg,

<    strlen(errmsg));

< } else

---

> /\* If error occurred on mux client, append to their stderr \*/

> if (tochan)

> buffer\_append(&c->extended, errmsg, strlen(errmsg));

> else

780,789c773

< if (cr->action == CONFIRM\_TTY) {

< /\*

< \* If a TTY allocation error occurred, then arrange

< \* for the correct TTY to leave raw mode.

< \*/

< if (c->self == session\_ident)

< leave\_raw\_mode(0);

< else

< mux\_tty\_alloc\_failed(c);

< } else if (cr->action == CONFIRM\_CLOSE) {

---

> if (cr->do\_close) {

803,805c787,788

< void

< client\_expect\_confirm(int id, const char \*request,

<     enum confirm\_action action)

---

> static void

> client\_expect\_confirm(int id, const char \*request, int do\_close)

810c793

< cr->action = action;

---

> cr->do\_close = do\_close;

850c833

< leave\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

954c937

< enter\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> enter\_raw\_mode(force\_tty\_flag);

1073,1074c1056

< leave\_raw\_mode(

<    options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

1369c1351

< leave\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

1440c1422

< enter\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> enter\_raw\_mode(force\_tty\_flag);

1444,1451c1426,1431

< if (session\_ident != -1) {

< if (escape\_char\_arg != SSH\_ESCAPECHAR\_NONE) {

< channel\_register\_filter(session\_ident,

<    client\_simple\_escape\_filter, NULL,

<    client\_filter\_cleanup,

<    client\_new\_escape\_filter\_ctx(

<    escape\_char\_arg));

< }

---

> if (escape\_char\_arg != SSH\_ESCAPECHAR\_NONE)

> channel\_register\_filter(session\_ident,

>    client\_simple\_escape\_filter, NULL,

>    client\_filter\_cleanup,

>    client\_new\_escape\_filter\_ctx(escape\_char\_arg));

> if (session\_ident != -1)

1454d1433

< }

1510a1490,1498

>

> #ifdef GSSAPI

> if (options.gss\_renewal\_rekey &&

>    ssh\_gssapi\_credentials\_updated(GSS\_C\_NO\_CONTEXT)) {

> debug("credentials updated - forcing rekey");

> need\_rekeying = 1;

> }

> #endif

>

1584c1572

< leave\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

2010c1998

< client\_expect\_confirm(id, "PTY allocation", CONFIRM\_TTY);

---

> client\_expect\_confirm(id, "PTY allocation", 1);

2069c2057

< client\_expect\_confirm(id, "subsystem", CONFIRM\_CLOSE);

---

> client\_expect\_confirm(id, "subsystem", 1);

2074c2062

< client\_expect\_confirm(id, "exec", CONFIRM\_CLOSE);

---

> client\_expect\_confirm(id, "exec", 1);

2080c2068

< client\_expect\_confirm(id, "shell", CONFIRM\_CLOSE);

---

> client\_expect\_confirm(id, "shell", 1);

2150,2164d2137

< void

< client\_stop\_mux(void)

< {

< if (options.control\_path != NULL && muxserver\_sock != -1)

< unlink(options.control\_path);

< /\*

< \* If we are in persist mode, signal that we should close when all

< \* active channels are closed.

< \*/

< if (options.control\_persist) {

< session\_closed = 1;

< setproctitle("[stopped mux]");

< }

< }

<

2169c2142

< leave\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

**gss-genr.c**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff gss-genr.c ../../moonshot/openssh/gss-genr.c

4c4

<  \* Copyright (c) 2001-2007 Simon Wilkinson. All rights reserved.

---

>  \* Copyright (c) 2001-2009 Simon Wilkinson. All rights reserved.

41a42,45

> #include "cipher.h"

> #include "key.h"

> #include "kex.h"

> #include <openssl/evp.h>

47a52,202

> typedef struct {

> char \*encoded;

> gss\_OID oid;

> } ssh\_gss\_kex\_mapping;

>

> /\*

>  \* XXX - It would be nice to find a more elegant way of handling the

>  \* XXX   passing of the key exchange context to the userauth routines

>  \*/

>

> Gssctxt \*gss\_kex\_context = NULL;

>

> static ssh\_gss\_kex\_mapping \*gss\_enc2oid = NULL;

>

> int

> ssh\_gssapi\_oid\_table\_ok() {

> return (gss\_enc2oid != NULL);

> }

>

> /\*

>  \* Return a list of the gss-group1-sha1 mechanisms supported by this program

>  \*

>  \* We test mechanisms to ensure that we can use them, to avoid starting

>  \* a key exchange with a bad mechanism

>  \*/

>

> char \*

> ssh\_gssapi\_client\_mechanisms(const char \*host, const char \*client) {

> gss\_OID\_set gss\_supported;

> OM\_uint32 min\_status;

>

> if (GSS\_ERROR(gss\_indicate\_mechs(&min\_status, &gss\_supported)))

> return NULL;

>

> return(ssh\_gssapi\_kex\_mechs(gss\_supported, ssh\_gssapi\_check\_mechanism,

>    host, client));

> }

>

> char \*

> ssh\_gssapi\_kex\_mechs(gss\_OID\_set gss\_supported, ssh\_gssapi\_check\_fn \*check,

>     const char \*host, const char \*client) {

> Buffer buf;

> size\_t i;

> int oidpos, enclen;

> char \*mechs, \*encoded;

> u\_char digest[EVP\_MAX\_MD\_SIZE];

> char deroid[2];

> const EVP\_MD \*evp\_md = EVP\_md5();

> EVP\_MD\_CTX md;

>

> if (gss\_enc2oid != NULL) {

> for (i = 0; gss\_enc2oid[i].encoded != NULL; i++)

> xfree(gss\_enc2oid[i].encoded);

> xfree(gss\_enc2oid);

> }

>

> gss\_enc2oid = xmalloc(sizeof(ssh\_gss\_kex\_mapping) \*

>    (gss\_supported->count + 1));

>

> buffer\_init(&buf);

>

> oidpos = 0;

> for (i = 0; i < gss\_supported->count; i++) {

> if (gss\_supported->elements[i].length < 128 &&

>    (\*check)(NULL, &(gss\_supported->elements[i]), host, client)) {

>

> deroid[0] = SSH\_GSS\_OIDTYPE;

> deroid[1] = gss\_supported->elements[i].length;

>

> EVP\_DigestInit(&md, evp\_md);

> EVP\_DigestUpdate(&md, deroid, 2);

> EVP\_DigestUpdate(&md,

>    gss\_supported->elements[i].elements,

>    gss\_supported->elements[i].length);

> EVP\_DigestFinal(&md, digest, NULL);

>

> encoded = xmalloc(EVP\_MD\_size(evp\_md) \* 2);

> enclen = \_\_b64\_ntop(digest, EVP\_MD\_size(evp\_md),

>    encoded, EVP\_MD\_size(evp\_md) \* 2);

>

> if (oidpos != 0)

> buffer\_put\_char(&buf, ',');

>

> buffer\_append(&buf, KEX\_GSS\_GEX\_SHA1\_ID,

>    sizeof(KEX\_GSS\_GEX\_SHA1\_ID) - 1);

> buffer\_append(&buf, encoded, enclen);

> buffer\_put\_char(&buf, ',');

> buffer\_append(&buf, KEX\_GSS\_GRP1\_SHA1\_ID,

>    sizeof(KEX\_GSS\_GRP1\_SHA1\_ID) - 1);

> buffer\_append(&buf, encoded, enclen);

> buffer\_put\_char(&buf, ',');

> buffer\_append(&buf, KEX\_GSS\_GRP14\_SHA1\_ID,

>    sizeof(KEX\_GSS\_GRP14\_SHA1\_ID) - 1);

> buffer\_append(&buf, encoded, enclen);

>

> gss\_enc2oid[oidpos].oid = &(gss\_supported->elements[i]);

> gss\_enc2oid[oidpos].encoded = encoded;

> oidpos++;

> }

> }

> gss\_enc2oid[oidpos].oid = NULL;

> gss\_enc2oid[oidpos].encoded = NULL;

>

> buffer\_put\_char(&buf, '\0');

>

> mechs = xmalloc(buffer\_len(&buf));

> buffer\_get(&buf, mechs, buffer\_len(&buf));

> buffer\_free(&buf);

>

> if (strlen(mechs) == 0) {

> xfree(mechs);

> mechs = NULL;

> }

>

> return (mechs);

> }

>

> gss\_OID

> ssh\_gssapi\_id\_kex(Gssctxt \*ctx, char \*name, int kex\_type) {

> int i = 0;

>

> switch (kex\_type) {

> case KEX\_GSS\_GRP1\_SHA1:

> if (strlen(name) < sizeof(KEX\_GSS\_GRP1\_SHA1\_ID))

> return GSS\_C\_NO\_OID;

> name += sizeof(KEX\_GSS\_GRP1\_SHA1\_ID) - 1;

> break;

> case KEX\_GSS\_GRP14\_SHA1:

> if (strlen(name) < sizeof(KEX\_GSS\_GRP14\_SHA1\_ID))

> return GSS\_C\_NO\_OID;

> name += sizeof(KEX\_GSS\_GRP14\_SHA1\_ID) - 1;

> break;

> case KEX\_GSS\_GEX\_SHA1:

> if (strlen(name) < sizeof(KEX\_GSS\_GEX\_SHA1\_ID))

> return GSS\_C\_NO\_OID;

> name += sizeof(KEX\_GSS\_GEX\_SHA1\_ID) - 1;

> break;

> default:

> return GSS\_C\_NO\_OID;

> }

>

> while (gss\_enc2oid[i].encoded != NULL &&

>    strcmp(name, gss\_enc2oid[i].encoded) != 0)

> i++;

>

> if (gss\_enc2oid[i].oid != NULL && ctx != NULL)

> ssh\_gssapi\_set\_oid(ctx, gss\_enc2oid[i].oid);

>

> return gss\_enc2oid[i].oid;

> }

>

200c355

<    GSS\_C\_NO\_CREDENTIAL, &ctx->context, ctx->name, ctx->oid,

---

>    ctx->client\_creds, &ctx->context, ctx->name, ctx->oid,

229a385,415

> ssh\_gssapi\_client\_identity(Gssctxt \*ctx, const char \*name)

> {

> gss\_buffer\_desc gssbuf;

> gss\_name\_t gssname;

> OM\_uint32 status;

> gss\_OID\_set oidset;

>

> gssbuf.value = (void \*) name;

> gssbuf.length = strlen(gssbuf.value);

>

> gss\_create\_empty\_oid\_set(&status, &oidset);

> gss\_add\_oid\_set\_member(&status, ctx->oid, &oidset);

>

> ctx->major = gss\_import\_name(&ctx->minor, &gssbuf,

>    GSS\_C\_NT\_USER\_NAME, &gssname);

>

> if (!ctx->major)

> ctx->major = gss\_acquire\_cred(&ctx->minor,

>    gssname, 0, oidset, GSS\_C\_INITIATE,

>    &ctx->client\_creds, NULL, NULL);

>

> gss\_release\_name(&status, &gssname);

> gss\_release\_oid\_set(&status, &oidset);

>

> if (ctx->major)

> ssh\_gssapi\_error(ctx);

>

> return(ctx->major);

> }

>

> OM\_uint32

231a418,420

> if (ctx == NULL)

> return -1;

>

238a428,440

> /\* Priviledged when used by server \*/

> OM\_uint32

> ssh\_gssapi\_checkmic(Gssctxt \*ctx, gss\_buffer\_t gssbuf, gss\_buffer\_t gssmic)

> {

> if (ctx == NULL)

> return -1;

>

> ctx->major = gss\_verify\_mic(&ctx->minor, ctx->context,

>    gssbuf, gssmic, NULL);

>

> return (ctx->major);

> }

>

252c454,455

< ssh\_gssapi\_check\_mechanism(Gssctxt \*\*ctx, gss\_OID oid, const char \*host)

---

> ssh\_gssapi\_check\_mechanism(Gssctxt \*\*ctx, gss\_OID oid, const char \*host,

>     const char \*client)

256a460,463

> Gssctxt \*intctx = NULL;

>

> if (ctx == NULL)

> ctx = &intctx;

265a473,476

>

> if (!GSS\_ERROR(major) && client)

> major = ssh\_gssapi\_client\_identity(\*ctx, client);

>

275c486

< if (GSS\_ERROR(major))

---

> if (GSS\_ERROR(major) || intctx != NULL)

280a492,548

> int

> ssh\_gssapi\_credentials\_updated(Gssctxt \*ctxt) {

> static gss\_name\_t saved\_name = GSS\_C\_NO\_NAME;

> static OM\_uint32 saved\_lifetime = 0;

> static gss\_OID saved\_mech = GSS\_C\_NO\_OID;

> static gss\_name\_t name;

> static OM\_uint32 last\_call = 0;

> OM\_uint32 lifetime, now, major, minor;

> int equal;

> gss\_cred\_usage\_t usage = GSS\_C\_INITIATE;

>

> now = time(NULL);

>

> if (ctxt) {

> debug("Rekey has happened - updating saved versions");

>

> if (saved\_name != GSS\_C\_NO\_NAME)

> gss\_release\_name(&minor, &saved\_name);

>

> major = gss\_inquire\_cred(&minor, GSS\_C\_NO\_CREDENTIAL,

>    &saved\_name, &saved\_lifetime, NULL, NULL);

>

> if (!GSS\_ERROR(major)) {

> saved\_mech = ctxt->oid;

>        saved\_lifetime+= now;

> } else {

> /\* Handle the error \*/

> }

> return 0;

> }

>

> if (now - last\_call < 10)

> return 0;

>

> last\_call = now;

>

> if (saved\_mech == GSS\_C\_NO\_OID)

> return 0;

>

> major = gss\_inquire\_cred(&minor, GSS\_C\_NO\_CREDENTIAL,

>    &name, &lifetime, NULL, NULL);

> if (major == GSS\_S\_CREDENTIALS\_EXPIRED)

> return 0;

> else if (GSS\_ERROR(major))

> return 0;

>

> major = gss\_compare\_name(&minor, saved\_name, name, &equal);

> gss\_release\_name(&minor, &name);

> if (GSS\_ERROR(major))

> return 0;

>

> if (equal && (saved\_lifetime < lifetime + now - 10))

> return 1;

>

> return 0;

> }

>

**kex.c**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff kex.c ../../moonshot/openssh/kex.c

52a53,56

> #ifdef GSSAPI

> #include "ssh-gss.h"

> #endif

>

360a365,378

> #ifdef GSSAPI

> } else if (strncmp(k->name, KEX\_GSS\_GEX\_SHA1\_ID,

>    sizeof(KEX\_GSS\_GEX\_SHA1\_ID) - 1) == 0) {

> k->kex\_type = KEX\_GSS\_GEX\_SHA1;

> k->evp\_md = EVP\_sha1();

> } else if (strncmp(k->name, KEX\_GSS\_GRP1\_SHA1\_ID,

>    sizeof(KEX\_GSS\_GRP1\_SHA1\_ID) - 1) == 0) {

> k->kex\_type = KEX\_GSS\_GRP1\_SHA1;

> k->evp\_md = EVP\_sha1();

> } else if (strncmp(k->name, KEX\_GSS\_GRP14\_SHA1\_ID,

>    sizeof(KEX\_GSS\_GRP14\_SHA1\_ID) - 1) == 0) {

> k->kex\_type = KEX\_GSS\_GRP14\_SHA1;

> k->evp\_md = EVP\_sha1();

> #endif

**kex.h**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff kex.h ../../moonshot/openssh/kex.h

75a76,78

> KEX\_GSS\_GRP1\_SHA1,

> KEX\_GSS\_GRP14\_SHA1,

> KEX\_GSS\_GEX\_SHA1,

131a135,140

> #ifdef GSSAPI

> int gss\_deleg\_creds;

> int gss\_trust\_dns;

> char    \*gss\_host;

> char \*gss\_client;

> #endif

158a168,172

> #ifdef GSSAPI

> void kexgss\_client(Kex \*);

> void kexgss\_server(Kex \*);

> #endif

>

**log.c**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff log.c ../../moonshot/openssh/log.c

1c1

< /\* $OpenBSD: log.c,v 1.42 2011/06/17 21:44:30 djm Exp $ \*/

---

> /\* $OpenBSD: log.c,v 1.41 2008/06/10 04:50:25 dtucker Exp $ \*/

59,60d58

< static log\_handler\_fn \*log\_handler;

< static void \*log\_handler\_ctx;

265,267d262

< log\_handler = NULL;

< log\_handler\_ctx = NULL;

<

335,351d329

< set\_log\_handler(log\_handler\_fn \*handler, void \*ctx)

< {

< log\_handler = handler;

< log\_handler\_ctx = ctx;

< }

<

< void

< do\_log2(LogLevel level, const char \*fmt,...)

< {

< va\_list args;

<

< va\_start(args, fmt);

< do\_log(level, fmt, args);

< va\_end(args);

< }

<

< void

362d339

< log\_handler\_fn \*tmp\_handler;

401c378

< if (txt != NULL && log\_handler == NULL) {

---

> if (txt != NULL) {

409,415c386

< if (log\_handler != NULL) {

< /\* Avoid recursion \*/

< tmp\_handler = log\_handler;

< log\_handler = NULL;

< tmp\_handler(level, fmtbuf, log\_handler\_ctx);

< log\_handler = tmp\_handler;

< } else if (log\_on\_stderr) {

---

> if (log\_on\_stderr) {

**monitor.h**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff monitor.h ../../moonshot/openssh/monitor.h

1c1

< /\* $OpenBSD: monitor.h,v 1.16 2011/06/17 21:44:31 djm Exp $ \*/

---

> /\* $OpenBSD: monitor.h,v 1.15 2008/11/04 08:22:13 djm Exp $ \*/

54a55

> MONITOR\_REQ\_GSSLOCALNAME, MONITOR\_ANS\_GSSLOCALNAME,

55a57,58

> MONITOR\_REQ\_GSSSIGN, MONITOR\_ANS\_GSSSIGN,

> MONITOR\_REQ\_GSSUPCREDS, MONITOR\_ANS\_GSSUPCREDS,

75,76d77

< int m\_log\_recvfd;

< int m\_log\_sendfd;

**mux.c**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff mux.c ../../moonshot/openssh/mux.c

1c1

< /\* $OpenBSD: mux.c,v 1.29 2011/06/22 22:08:42 djm Exp $ \*/

---

> /\* $OpenBSD: mux.c,v 1.24 2011/01/13 21:54:53 djm Exp $ \*/

89a90

> extern int force\_tty\_flag;

148d148

< #define MUX\_C\_STOP\_LISTENING 0x10000009

156d155

< #define MUX\_S\_TTY\_ALLOC\_FAIL 0x80000008

172d170

< static int process\_mux\_stop\_listening(u\_int, Channel \*, Buffer \*, Buffer \*);

185d182

< { MUX\_C\_STOP\_LISTENING, process\_mux\_stop\_listening },

921,953d917

< static int

< process\_mux\_stop\_listening(u\_int rid, Channel \*c, Buffer \*m, Buffer \*r)

< {

< debug("%s: channel %d: stop listening", \_\_func\_\_, c->self);

<

< if (options.control\_master == SSHCTL\_MASTER\_ASK ||

<    options.control\_master == SSHCTL\_MASTER\_AUTO\_ASK) {

< if (!ask\_permission("Disable further multiplexing on shared "

<    "connection to %s? ", host)) {

< debug2("%s: stop listen refused by user", \_\_func\_\_);

< buffer\_put\_int(r, MUX\_S\_PERMISSION\_DENIED);

< buffer\_put\_int(r, rid);

< buffer\_put\_cstring(r, "Permission denied");

< return 0;

< }

< }

<

< if (mux\_listener\_channel != NULL) {

< channel\_free(mux\_listener\_channel);

< client\_stop\_mux();

< xfree(options.control\_path);

< options.control\_path = NULL;

< mux\_listener\_channel = NULL;

< muxserver\_sock = -1;

< }

<

< /\* prepare reply \*/

< buffer\_put\_int(r, MUX\_S\_OK);

< buffer\_put\_int(r, rid);

<

< return 0;

< }

<

1058,1078d1021

< void

< mux\_tty\_alloc\_failed(Channel \*c)

< {

< Buffer m;

< Channel \*mux\_chan;

<

< debug3("%s: channel %d: TTY alloc failed", \_\_func\_\_, c->self);

<

< if ((mux\_chan = channel\_by\_id(c->ctl\_chan)) == NULL)

< fatal("%s: channel %d missing mux channel %d",

<    \_\_func\_\_, c->self, c->ctl\_chan);

<

< /\* Append exit message packet to control socket output queue \*/

< buffer\_init(&m);

< buffer\_put\_int(&m, MUX\_S\_TTY\_ALLOC\_FAIL);

< buffer\_put\_int(&m, c->self);

<

< buffer\_put\_string(&mux\_chan->output, buffer\_ptr(&m), buffer\_len(&m));

< buffer\_free(&m);

< }

<

1119,1123c1062,1063

<    sizeof(addr.sun\_path)) >= sizeof(addr.sun\_path)) {

< error("ControlPath \"%s\" too long for Unix domain socket",

<    options.control\_path);

< goto disable\_mux\_master;

< }

---

>    sizeof(addr.sun\_path)) >= sizeof(addr.sun\_path))

> fatal("ControlPath too long");

1129a1070

> muxserver\_sock = -1;

1134,1137c1075,1076

< if (muxserver\_sock != -1) {

< close(muxserver\_sock);

< muxserver\_sock = -1;

< }

---

> close(muxserver\_sock);

> muxserver\_sock = -1;

1217,1220c1156,1157

< x11\_request\_forwarding\_with\_spoofing(id, display, proto,

<    data, 1);

< client\_expect\_confirm(id, "X11 forwarding", CONFIRM\_WARN);

< /\* XXX exit\_on\_forward\_failure \*/

---

> x11\_request\_forwarding\_with\_spoofing(id, display, proto, data);

> /\* XXX wait for reply \*/

1639c1576

< int devnull, rawmode;

---

> int devnull;

1735d1671

< rawmode = tty\_flag;

1737c1673

< enter\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> enter\_raw\_mode(force\_tty\_flag);

1751,1771c1687

< switch (type) {

< case MUX\_S\_TTY\_ALLOC\_FAIL:

< if ((esid = buffer\_get\_int(&m)) != sid)

< fatal("%s: tty alloc fail on unknown session: "

<    "my id %u theirs %u",

<    \_\_func\_\_, sid, esid);

< leave\_raw\_mode(options.request\_tty ==

<    REQUEST\_TTY\_FORCE);

< rawmode = 0;

< continue;

< case MUX\_S\_EXIT\_MESSAGE:

< if ((esid = buffer\_get\_int(&m)) != sid)

< fatal("%s: exit on unknown session: "

<    "my id %u theirs %u",

<    \_\_func\_\_, sid, esid);

< if (exitval\_seen)

< fatal("%s: exitval sent twice", \_\_func\_\_);

< exitval = buffer\_get\_int(&m);

< exitval\_seen = 1;

< continue;

< default:

---

> if (type != MUX\_S\_EXIT\_MESSAGE) {

1774a1691,1698

> if ((esid = buffer\_get\_int(&m)) != sid)

> fatal("%s: exit on unknown session: my id %u theirs %u",

>    \_\_func\_\_, sid, esid);

> debug("%s: master session id: %u", \_\_func\_\_, sid);

> if (exitval\_seen)

> fatal("%s: exitval sent twice", \_\_func\_\_);

> exitval = buffer\_get\_int(&m);

> exitval\_seen = 1;

1778,1779c1702

< if (rawmode)

< leave\_raw\_mode(options.request\_tty == REQUEST\_TTY\_FORCE);

---

> leave\_raw\_mode(force\_tty\_flag);

1893,1936d1815

< static void

< mux\_client\_request\_stop\_listening(int fd)

< {

< Buffer m;

< char \*e;

< u\_int type, rid;

<

< debug3("%s: entering", \_\_func\_\_);

<

< buffer\_init(&m);

< buffer\_put\_int(&m, MUX\_C\_STOP\_LISTENING);

< buffer\_put\_int(&m, muxclient\_request\_id);

<

< if (mux\_client\_write\_packet(fd, &m) != 0)

< fatal("%s: write packet: %s", \_\_func\_\_, strerror(errno));

<

< buffer\_clear(&m);

<

< /\* Read their reply \*/

< if (mux\_client\_read\_packet(fd, &m) != 0)

< fatal("%s: read from master failed: %s",

<    \_\_func\_\_, strerror(errno));

<

< type = buffer\_get\_int(&m);

< if ((rid = buffer\_get\_int(&m)) != muxclient\_request\_id)

< fatal("%s: out of sequence reply: my id %u theirs %u",

<    \_\_func\_\_, muxclient\_request\_id, rid);

< switch (type) {

< case MUX\_S\_OK:

< break;

< case MUX\_S\_PERMISSION\_DENIED:

< e = buffer\_get\_string(&m, NULL);

< fatal("Master refused stop listening request: %s", e);

< case MUX\_S\_FAILURE:

< e = buffer\_get\_string(&m, NULL);

< fatal("%s: stop listening request failed: %s", \_\_func\_\_, e);

< default:

< fatal("%s: unexpected response from master 0x%08x",

<    \_\_func\_\_, type);

< }

< buffer\_free(&m);

< muxclient\_request\_id++;

< }

<

2030,2033d1908

< case SSHMUX\_COMMAND\_STOP:

< mux\_client\_request\_stop\_listening(sock);

< fprintf(stderr, "Stop listening request sent.\r\n");

< exit(0);

**PROTOCOL.mux**

dhcp-134-127:openssh-5.9p1 koenveelenturf$ diff PROTOCOL.mux ../../moonshot/openssh/PROTOCOL.mux

76,82d75

< A master may also send a MUX\_S\_TTY\_ALLOC\_FAIL before MUX\_S\_EXIT\_MESSAGE

< if remote TTY allocation was unsuccessful. The client may use this to

< return its local tty to "cooked" mode.

<

< uint32 MUX\_S\_TTY\_ALLOC\_FAIL

< uint32 session id

<

159c152

< A server may reply with a MUX\_S\_SESSION\_OPENED, a MUX\_S\_PERMISSION\_DENIED

---

> A server may reply with a MUX\_S\_SESSION\_OPEED, a MUX\_S\_PERMISSION\_DENIED

162,173c155

< 8. Requesting shutdown of mux listener

<

< A client may request the master to stop accepting new multiplexing requests

< and remove its listener socket.

<

< uint32 MUX\_C\_STOP\_LISTENING

< uint32 request id

<

< A server may reply with a MUX\_S\_OK, a MUX\_S\_PERMISSION\_DENIED or a

< MUX\_S\_FAILURE.

<

< 9. Status messages

---

> 8. Status messages

190c172

< 10. Protocol numbers

---

> 9. Protocol numbers

199d180

< #define MUX\_C\_STOP\_LISTENING 0x10000009

207d187

< #define MUX\_S\_TTY\_ALLOC\_FAIL 0x80000008

214a195

> XXX graceful close (delete listening socket, but keep existing sessions active)

218a200

> XXX port0 rfwd (need custom response message)

221c203

< $OpenBSD: PROTOCOL.mux,v 1.7 2011/05/08 12:52:01 djm Exp $

---

> $OpenBSD: PROTOCOL.mux,v 1.4 2011/01/31 21:42:15 djm Exp $