```
**********************************************************
    2350 Fri Mar 14 22:43:08 2014
new/usr/src/lib/pkcs11/pkcs11_tpm/Makefile.com
XXXX pkcs11_tpm blithely connects with TCP when it shouldn't.
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
LIBRARY =       pkcs11_tpm.a
VERS =          .1

OBJECTS= api_interface.o \
        apiutil.o \
        asn1.o \
        cert.o \
        data_obj.o \
        decr_mgr.o \
        dig_mgr.o \
        encr_mgr.o \
        globals.o \
        hwf_obj.o \
        key.o \
        key_mgr.o \
        loadsave.o \
        log.o \
        mech_md5.o \
        mech_rsa.o \
        mech_sha.o \
        new_host.o \
        obj_mgr.o \
        object.o \
        sess_mgr.o \
        sign_mgr.o \
        template.o \
        tpm_specific.o \
        utility.o \
        verify_mgr.o


include $(SRC)/lib/Makefile.lib

SRCDIR= ../common

SRCS=   $(OBJECTS:%.o=$(SRCDIR)/%.c)

#       set signing mode
```



```
POST_PROCESS_SO +=      ; $(ELFSIGN_CRYPTO)

ROOTLIBDIR=$(ROOT)/usr/lib/security
ROOTLIBDIR64=$(ROOT)/usr/lib/security/$(MACH64)

LIBS=$(DYNLIB) $(DYNLIB64)

TSSROOT=$(ADJUNCT_PROTO)
TSPILIBDIR=$(TSSROOT)/usr/lib
TSPIINCDIR=$(TSSROOT)/usr/include
TSSLIB=-L$(TSPILIBDIR)
TSSLIB64=-L$(TSPILIBDIR)/$(MACH64)
TSSINC=-I$(TSPIINCDIR)

LDLIBS += $(TSSLIB) -L$(ADJUNCT_PROTO)/lib -lc -luuid -lmd -ltspi -lcrypto -lscf
LDLIBS += $(TSSLIB) -L$(ADJUNCT_PROTO)/lib -lc -luuid -lmd -ltspi -lcrypto
CPPFLAGS += -xCC -D_POSIX_PTHREAD_SEMANTICS $(TSSINC)
CPPFLAGS64 += $(CPPFLAGS)
C99MODE=        $(C99_ENABLE)

CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-unused-label
CERRWARN +=     -_gcc=-Wno-uninitialized

LINTSRC= $(OBJECTS:%.o=$(SRCDIR)/%.c)

$(LINTLIB):=    SRCS    =       $(SRCDIR)/$(LINTSRC)
LINTSRC= $(SRCS)

CLOBBERFILES += C.ln

.KEEP_STATE:

all: $(LIBS)

lint: $$(LINTSRC)
        $(LINT.c) $(LINTCHECKFLAGS) $(LINTSRC) $(LDLIBS)

pics/%.o: $(SRCDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

include $(SRC)/lib/Makefile.targ
```

```
**********************************************************
   72374 Fri Mar 14 22:43:09 2014
new/usr/src/lib/pkcs11/pkcs11_tpm/common/tpm_specific.c
XXXX pkcs11_tpm blithely connects with TCP when it shouldn't.
**********************************************************
```

```
/*
 * The Initial Developer of the Original Code is International
 * Business Machines Corporation. Portions created by IBM
 * Corporation are Copyright (C) 2005 International Business
 * Machines Corporation. All Rights Reserved.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the Common Public License as published by
 * IBM Corporation; either version 1 of the License, or (at your option)
 * any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * Common Public License for more details.
 *
 * You should have received a copy of the Common Public License
 * along with this program; if not, a copy can be viewed at
 * http://www.opensource.org/licenses/cpl1.0.php.
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2012 Milan Jurik. All rights reserved.
 */

#include <pthread.h>
#include <string.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <uuid/uuid.h>
#include <fcntl.h>
#include <errno.h>
#include <pwd.h>
#include <syslog.h>
#include <libscf.h>

#include <openssl/rsa.h>

#include <tss/platform.h>
#include <tss/tss_defines.h>
#include <tss/tss_typedef.h>
#include <tss/tss_structs.h>
#include <tss/tss_error.h>
#include <tss/tcs_error.h>
#include <tss/tspi.h>
#include <trousers/trousers.h>

#include "tpmtok_int.h"
#include "tpmtok_defs.h"

#define MAX_RSA_KEYLENGTH 512

extern void stlogit(char *fmt, ...);

CK_RV token_rng(TSS_HCONTEXT, CK_BYTE *,  CK_ULONG);
int tok_slot2local(CK_SLOT_ID);
CK_RV token_specific_session(CK_SLOT_ID);
CK_RV token_specific_final(TSS_HCONTEXT);
```



```
CK_RV
token_specific_rsa_decrypt(
        TSS_HCONTEXT,
        CK_BYTE *,
        CK_ULONG,
        CK_BYTE *,
        CK_ULONG *,
        OBJECT *);

CK_RV
token_specific_rsa_encrypt(
        TSS_HCONTEXT,
        CK_BYTE *,
        CK_ULONG,
        CK_BYTE *,
        CK_ULONG *,
        OBJECT *);

CK_RV
token_specific_rsa_sign(
        TSS_HCONTEXT,
        CK_BYTE *,
        CK_ULONG,
        CK_BYTE *,
        CK_ULONG *,
        OBJECT *);

CK_RV
token_specific_rsa_verify(TSS_HCONTEXT, CK_BYTE *,
    CK_ULONG, CK_BYTE *, CK_ULONG, OBJECT *);

CK_RV
token_specific_rsa_generate_keypair(TSS_HCONTEXT,
        TEMPLATE *,
        TEMPLATE *);

CK_RV
token_specific_sha_init(DIGEST_CONTEXT *);

CK_RV
token_specific_sha_update(DIGEST_CONTEXT *,
        CK_BYTE *,
        CK_ULONG);

CK_RV
token_specific_sha_final(DIGEST_CONTEXT *,
        CK_BYTE *,
        CK_ULONG *);

CK_RV token_specific_login(TSS_HCONTEXT, CK_USER_TYPE, CK_CHAR_PTR, CK_ULONG);
CK_RV token_specific_logout(TSS_HCONTEXT);
CK_RV token_specific_init_pin(TSS_HCONTEXT, CK_CHAR_PTR, CK_ULONG);
CK_RV token_specific_set_pin(ST_SESSION_HANDLE, CK_CHAR_PTR,
        CK_ULONG, CK_CHAR_PTR, CK_ULONG);
CK_RV token_specific_verify_so_pin(TSS_HCONTEXT, CK_CHAR_PTR, CK_ULONG);

static CK_RV
token_specific_init(char *, CK_SLOT_ID, TSS_HCONTEXT *);

struct token_specific_struct token_specific = {
        "TPM_Debug",
        &token_specific_init,
        NULL,
        &token_rng,
        &token_specific_session,
        &token_specific_final,
```

```
 128         &token_specific_rsa_decrypt,
 129         &token_specific_rsa_encrypt,
 130         &token_specific_rsa_sign,
 131         &token_specific_rsa_verify,
 132         &token_specific_rsa_generate_keypair,
 133         NULL,
 134         NULL,
 135         NULL,
 136         &token_specific_login,
 137         &token_specific_logout,
 138         &token_specific_init_pin,
 139         &token_specific_set_pin,
 140         &token_specific_verify_so_pin
 141 };
______unchanged_portion_omitted_

 625 TSS_RESULT
 626 open_tss_context(TSS_HCONTEXT *pContext)
 627 {
 628         TSS_RESULT result;
 629         char *smf_string;

 631         /*
 632          * The Tspi_* functions fail if we don't have tcsd running.  Worse,
 633          * because Tspi_* uses TCP over localhost, this can accidentally
 634          * trigger anti-denial-of-service measures.
 635          *
 636          * Instead, use SMF to see if tcsd is fully up and running, or if it
 637          * exists at all.  If it's not, bail early.
 638          */
 639         /* XXX KEBE ASKS -> Use a hardwired FMRI string or instance here? */
 640         smf_string = smf_get_state("svc:/application/security/tcsd:default");
 641         if (smf_string == NULL ||
 642             strcmp(smf_string, SCF_STATE_STRING_ONLINE) != 0) {
 643                 free(smf_string);
 644                 return (CKR_FUNCTION_FAILED);
 645         }
 646         free(smf_string);

 648         if ((result = Tspi_Context_Create(pContext))) {
 649                 stlogit("Tspi_Context_Create: 0x%0x - %s",
 650                     result, Trspi_Error_String(result));
 651                 return (CKR_FUNCTION_FAILED);
 652         }

 654         if ((result = Tspi_Context_Connect(*pContext, NULL))) {
 655                 stlogit("Tspi_Context_Connect: 0x%0x - %s",
 656                     result, Trspi_Error_String(result));
 657                 Tspi_Context_Close(*pContext);
 658                 *pContext = 0;
 659                 return (CKR_FUNCTION_FAILED);
 660         }
 661         return (result);
 662 }
______unchanged_portion_omitted_

2269 static CK_RV
2270 token_rsa_load_key(
2271         TSS_HCONTEXT hContext,
2272         OBJECT *key_obj,
2273         TSS_HKEY *phKey)
2274 {
2275         TSS_RESULT result;
2276         TSS_HPOLICY hPolicy = NULL_HPOLICY;
2277         TSS_HKEY        hParentKey;
2278         BYTE            *authData = NULL;
```



```
2279         CK_ATTRIBUTE    *attr;
2280         CK_RV           rc;
2281         CK_OBJECT_HANDLE handle;
2282         CK_ULONG        class;

2284         if (hPrivateLeafKey != NULL_HKEY) {
2285                 hParentKey = hPrivateRootKey;
2286         } else {
2287                 if ((result = token_load_public_root_key(hContext)))
2288                         return (CKR_FUNCTION_FAILED);

2290                 hParentKey = hPublicRootKey;
2291         }

2293         *phKey = (TSS_HKEY)(uintptr_t)NULL;
2274         *phKey = NULL;
2294         if (template_attribute_find(key_obj->template, CKA_CLASS,
2295             &attr) == FALSE) {
2296                 return (CKR_TEMPLATE_INCOMPLETE);
2297         }
2298         class = *((CK_ULONG *)attr->pValue);

2300         rc = template_attribute_find(key_obj->template,
2301             CKA_IBM_OPAQUE, &attr);
2302         /*
2303          * A public key cannot use the OPAQUE data attribute so they
2304          * must be created in software.  A private key may not yet
2305          * have its "opaque" data defined and needs to be created
2306          * and loaded so it can be used inside the TPM.
2307          */
2308         if (class == CKO_PUBLIC_KEY || rc == FALSE) {
2309                 rc = object_mgr_find_in_map2(hContext, key_obj, &handle);
2310                 if (rc != CKR_OK)
2311                         return (CKR_FUNCTION_FAILED);

2313                 if ((rc = token_load_key(hContext,
2314                     handle, hParentKey, NULL, phKey))) {
2315                         return (rc);
2316                 }
2317         }
2318         /*
2319          * If this is a private key, get the blob and load it in the TPM.
2320          * If it is public, the key is already loaded in software.
2321          */
2322         if (class == CKO_PRIVATE_KEY) {
2323                 /* If we already have a handle, just load it */
2324                 if (*phKey != (TSS_HKEY)(uintptr_t)NULL) {
2305                 if (*phKey != NULL) {
2325                         result = Tspi_Key_LoadKey(*phKey, hParentKey);
2326                         if (result) {
2327                                 stlogit("Tspi_Context_LoadKeyByBlob: "
2328                                     "0x%0x - %s",
2329                                     result, Trspi_Error_String(result));
2330                                 return (CKR_FUNCTION_FAILED);
2331                         }
2332                 } else {
2333                         /* try again to get the CKA_IBM_OPAQUE attr */
2334                         if ((rc = template_attribute_find(key_obj->template,
2335                             CKA_IBM_OPAQUE, &attr)) == FALSE) {
2336                                 return (rc);
2337                         }
2338                         if ((result = Tspi_Context_LoadKeyByBlob(hContext,
2339                             hParentKey, attr->ulValueLen, attr->pValue,
2340                             phKey))) {
2341                                 stlogit("Tspi_Context_LoadKeyByBlob: "
2342                                     "0x%0x - %s",
```

```
2343                                    result, Trspi_Error_String(result));
2344                                return (CKR_FUNCTION_FAILED);
2345                        }
2346                }
2347        }

2349        /* auth data may be required */
2350        if (template_attribute_find(key_obj->template, CKA_ENC_AUTHDATA,
2351            &attr) == TRUE && attr) {
2352                if ((hPrivateLeafKey == NULL_HKEY) &&
2353                    (hPublicLeafKey == NULL_HKEY)) {
2354                        return (CKR_FUNCTION_FAILED);
2355                } else if (hPublicLeafKey != NULL_HKEY) {
2356                        hParentKey = hPublicLeafKey;
2357                } else {
2358                        hParentKey = hPrivateLeafKey;
2359                }

2361                if ((result = token_unwrap_auth_data(hContext,
2362                    attr->pValue, attr->ulValueLen,
2363                    hParentKey, &authData))) {
2364                        return (CKR_FUNCTION_FAILED);
2365                }

2367                if ((result = Tspi_GetPolicyObject(*phKey,
2368                    TSS_POLICY_USAGE, &hPolicy))) {
2369                        stlogit("Tspi_GetPolicyObject: 0x%0x - %s",
2370                            result, Trspi_Error_String(result));
2371                        return (CKR_FUNCTION_FAILED);
2372                }

2374                /*
2375                 * If the policy handle returned is the same as the
2376                 * context's default policy, then a new policy must
2377                 * be created and assigned to the key. Otherwise, just set the
2378                 * secret in the policy.
2379                 */
2380                if (hPolicy == hDefaultPolicy) {
2381                        if ((result = Tspi_Context_CreateObject(hContext,
2382                            TSS_OBJECT_TYPE_POLICY, TSS_POLICY_USAGE,
2383                            &hPolicy))) {
2384                                stlogit("Tspi_Context_CreateObject: "
2385                                    "0x%0x - %s",
2386                                    result, Trspi_Error_String(result));
2387                                return (CKR_FUNCTION_FAILED);
2388                        }

2390                        if ((result = Tspi_Policy_SetSecret(hPolicy,
2391                            TSS_SECRET_MODE_SHA1,
2392                            SHA1_DIGEST_LENGTH, authData))) {
2393                                stlogit("Tspi_Policy_SetSecret: "
2394                                    "0x%0x - %s",
2395                                    result, Trspi_Error_String(result));
2396                                return (CKR_FUNCTION_FAILED);
2397                        }

2399                        if ((result = Tspi_Policy_AssignToObject(hPolicy,
2400                            *phKey))) {
2401                                stlogit("Tspi_Policy_AssignToObject: "
2402                                    "0x%0x - %s",
2403                                    result, Trspi_Error_String(result));
2404                                return (CKR_FUNCTION_FAILED);
2405                        }
2406                } else if ((result = Tspi_Policy_SetSecret(hPolicy,
2407                    TSS_SECRET_MODE_SHA1, SHA1_DIGEST_LENGTH, authData))) {
2408                        stlogit("Tspi_Policy_SetSecret: 0x%0x - %s",
```



```
2409                                    result, Trspi_Error_String(result));
2410                                return (CKR_FUNCTION_FAILED);
2411                }

2413                Tspi_Context_FreeMemory(hContext, authData);
2414        }

2416        return (CKR_OK);
2417 }
```

_____***unchanged_portion_omitted_***