**Beaglebone Black**

**Update kernel to bone-debian-8.3-lxqt-4gb-armhf-2016-01-24-4gb.img**

[**https://debian.beagleboard.org/images/bone-debian-8.3-lxqt-4gb-armhf-2016-01-24-4gb.img.xz**](https://debian.beagleboard.org/images/bone-debian-8.3-lxqt-4gb-armhf-2016-01-24-4gb.img.xz)

**Run these commands to update your packages to the latest versions:**

apt-get update

apt-get upgrade

**Run this command to install the packages needed going forward:**

apt-get install heimdal-clients heimdal-kcm krb5-config libkrb5-26-heimdal ssh git heimdal-dev libsasl2-modules-gssapi-heimdal git flex bison original-awk dh-autoreconf libncurses5-dev texinfo libXt-dev gcc make samba

**KRB5 config file**

**Edit your kerberos config file to match this, but change the hardcoded domain/realm info to what your domain/realm config is. This file controls how kerberos behaves.**

**/etc/krb5.conf**

[logging]

default=STDERR

[libdefaults]

default\_realm = CORPB.EXAMPLE.COM

dns\_lookup\_realm = false

dns\_lookup\_kdc = true

enable-pkinit = true

pkinit\_dh\_min\_bits = 1024

[realms]

CORPB.EXAMPLE.COM = {

pkinit\_require\_eku = true

pkinit\_require\_krbtgt\_otherName = true

auth\_to\_EXAMPLE.COM = RULE:[1:CORPB\$1]

kdc = SUBB.CORPB.EXAMPLE.COM

pkinit\_anchors = FILE:/home/CAhx/subb/cacert.pem

pkinit\_identities = FILE:/home/CAhx/subb/kdc.pem

enable-pkinit = true

}

CORPA.EXAMPLE.COM = {

pkinit\_require\_eku = true

pkinit\_require\_krbtgt\_otherName = true

auth\_to\_EXAMPLE.COM = RULE:[1:CORPA\$1]

kdc = SUBA.CORPA.EXAMPLE.COM

pkinit\_anchors = FILE:/home/CAhx/suba/cacert.pem

pkinit\_identities = FILE:/home/CAhx/suba/kdc.pem

enable-pkinit = true

}

[capaths]

CORPB.EXAMPLE.COM = {

CORPA.EXAMPLE.COM = CORPB.EXAMPLE.COM

}

[domain\_realm]

.corpb.example.com = CORPB.EXAMPLE.COM

corpb.example.com = CORPB.EXAMPLE.COM

.corpa.example.com = CORPA.EXAMPLE.COM

corpa.example.com = CORPA.EXAMPLE.COM

**SAMBA**

**Edit your samba config file to match this, but change the hardcoded domain/realm info to what your domain/realm config is. This file controls how samba behaves.**

**/etc/samba/smb.conf**

[global]

workgroup = CORPB

realm = CORPB.EXAMPLE.COM

kerberos method = system keytab

security = ads

client use spnego = yes

interfaces = eth0

winbind refresh tickets = yes

winbind use default domain = yes

allow trusted domains = yes

tls enabled = true

[netlogon]

path = /var/lib/samba/sysvol/corpb.example.com/scripts

read only = No

[sysvol]

path = /var/lib/samba/sysvol

read only = No

**Then run:**

/etc/init.d/samba restart

**To attempt to join the domain, use this command:**

net ads join -k

**Useful link for configuring PKINIT in Samba, if you need it:**

<https://wiki.samba.org/index.php/Samba_AD_Smart_Card_Login#Edit_the_Samba_KDC_Configuration_File_to_Enable_PKINIT_Authentication>

**Network config files**

**Edit your network config files to match this, but change the hardcoded ip/gateway/dns info to what your domain/realm config is. These files control how the networking on the machine works.**

**/etc/resolv.conf**

nameserver 172.17.100.33

search example.com

**/etc/network/interfaces**

auto lo

iface lo inet loopback

auto eth0

iface eth0 inet static

address 172.17.0.37

netmask 255.255.0.0

gateway 172.17.255.254

mtu 1462

up route add -net 10.1.1.0 netmask 255.255.255.0 gw 172.17.100.32 dev eth0

iface usb0 inet static

address 192.168.7.2

netmask 255.255.255.252

network 192.168.7.0

gateway 192.168.7.1

**/etc/hosts**

127.0.0.1 localhost

127.0.1.1 beaglebone3.corpb.example.com beaglebone3

172.17.0.36 SUBA.corpa.example.com SUBA

172.17.0.37 beaglebone3.corpa.example.com beaglebone3

172.17.0.39 beaglebone1.corpa.example.com beaglebone1

172.17.0.40 beaglebone2.corpc.example.com beaglebone2

10.1.1.13 SUBB.corpb.example.com SUBB

**Creating certs using custom private key (pr.pem) - For PKINIT**

**This demonstrates how to create the necessary files for PKINIT and how to test it.**

**--- USER ---**

**Remove passphrase**

openssl rsa -in pr.pem -out key.pem

RSA KEY -> CSR (Certificate Signing Request, by user, declaring what user)

openssl req -out csr.csr -key key.pem -new -subj "/UID=root/DC=dtkm/DC=net"

**Send CSR to AD DC**

**--- AD DC ---**

**Verify CSR - read what the CSR contains**

openssl req -text -noout -verify -in csr.csr

CSR -> CRT (CA sign the CSR)

openssl x509 -req -in csr.csr -CA ca.pem -out crt.crt -CAcreateserial -CAserial ca.seq -CA ca.pem

**Send CRT back to User**

scp crt.crt root@130.20.79.15:/home/heimdal

**--- USER ---**

**Combine crt/key into pem file - optional**

touch user.pem

cat crt.crt>>user.pem

cat key.pem>>user.pem

**Pkinit**

kinit -C FILE:user.pem

**OR, without combining files**

kinit -C FILE:crt.crt,key.pem

**Setting up the keytabs**

**Keytabs need to be exported in order to make sure permissions are properly granted for certain users to access certain services. Here is an example:**

**Create spn rcmd/beaglebone1.dtkm.local and rcmd/beaglebone1 for user BEAGLEBONE1$**

**(this is done on the AD-DC)**

samba-tool spn add rcmd/beaglebone1.dtkm.local BEAGLEBONE1$

samba-tool spn add rcmd/beaglebone1 BEAGLEBONE1$

**Export keytab**

samba-tool domain exportkeytab mykeytab-1 --principal=rcmd/beaglebone1.dtkm.local

samba-tool domain exportkeytab mykeytab-1 --principal=rcmd/beaglebone1

**Move to BBB, merge with /etc/krb5.keytab**

ktutil copy mykeytab-1 /etc/krb5.keytab

**run**

kinit   
net ads join -k

**To check, run**ktutil -k /etc/krb5.keytab list

**Ubuntu 16.04 Machine   
(Laptop with x86\_64 architecture in my case)**

apt-get update

apt-get upgrade

**Run this command to install all packages needed:**

apt-get install heimdal-clients heimdal-kcm krb5-config libkrb5-26-heimdal ssh git heimdal-dev libsasl2-modules-gssapi-heimdal git flex bison original-awk dh-autoreconf libncurses5-dev texinfo libxt-dev gcc make samba smbclient attr build-essential libacl1-dev libattr1-dev libblkid-dev libgnutls-dev libreadline-dev python-dev libpam0g-dev python-dnspython gdb pkg-config libpopt-dev libldap2-dev dnsutils libbsd-dev attr heimdal-clients docbook-xsl libcups2-dev acl winbind samba-dsdb-modules samba-vfs-modules -y

**To set up active directory with Samba, these instructions can be followed loosely:**

<https://wiki.samba.org/index.php/Setup_a_Samba_Active_Directory_Domain_Controller>

**Run this command to provision the domain:**

apt-get install winbind

samba-tool domain provision --use-rfc2307 --interactive

**The page above goes over the prerequisites for provisioning as well as how to verify things are working after provisioning.**

**Edit Kerberos file**

**Edit your kerberos config file to match this, but change the hardcoded domain/realm info to what your domain/realm config is. This file controls how kerberos behaves.**

**/etc/krb5.conf**

**(also make sure this file is in sync with /var/lib/samba/private/krb5.conf, use** ln -s**)**

File contents

**Edit Avahi Daemon File**

**Doing this fixed a weird issue where things were broken, it may help or not**

**edit /etc/avahi/avahi-daemon.conf:**

[server]

domain-name=.alocal

**Edit SAMBA File**

**Edit your samba config file to match this, but change the hardcoded domain/realm info to what your domain/realm config is. This file controls how samba behaves.**

**/etc/samba/smb.conf**

File contents

**Use** samba-tool **when you need to interface with samba. It handles a lot, from user management, to creating spns, to exporting keytabs, etc.**

**Network config files**

**Edit your network config files to match this, but change the hardcoded ip/gateway/dns info to what your domain/realm config is. These files control how the networking on the machine works.**

**/etc/resolv.conf**

File contents

**/etc/network/interfaces**

File contents

**/etc/hosts**

File contents

**Set up DNS settings in Ubuntu GUI top right corner**

Left-click Network Manager icon from the System Tray

Click “Edit Connections” from the menu

Select the appropriate tab (“Wired/Wireless“) depending on your connection

Double-click your connection

Select the IPv4 or IPv6 tab depending on your connection (if you’re not sure, you’re probably using IPv4)

Change the “Method” to “Automatic (DHCP) addresses only”

In the DNS servers box, enter 127.0.0.1

Click “Save“

Disconnect and reconnect to your network

**Edit NetworkManager File  
Disabling the NetworkManager may stop issues from occurring in the future, where it might change network config settings or network behavior without you knowing.**

**/etc/NetworkManager/NetworkManager.conf**

**comment out:**

#dns=dnsmasq

**Creating CA certs - For PKINIT**

**CA certs are used in the PKINIT process. Follow these commands to set them up properly.**

**root CA**

hxtool issue-certificate \

--self-signed \

--issue-ca \

--generate-key=rsa \

--subject="CN=CA,DC=corpb,DC=example,DC=com" \

--lifetime=10years \

--certificate="FILE:ca.pem"

**Identity / kdc.pem**

hxtool issue-certificate \

--ca-certificate=FILE:ca.pem \

--generate-key=rsa \

--type="pkinit-kdc" \

--pk-init-principal="krbtgt/CORPB.EXAMPLE.COM@CORPB.EXAMPLE.COM" \

--subject="uid=kdc,DC=corpb,DC=example,DC=com" \

--certificate="FILE:kdc.pem"

hxtool crl-sign \

--crl-file=crl.der \

--signer=FILE:ca.pem

**Creating user cert, or refer to other section above**

hxtool issue-certificate --ca-certificate=FILE:ca.pem --generate-key=rsa --type="pkinit-client" --pk-init-principal="root@CORPB.EXAMPLE.COM" --subject="uid=root,DC=corpb,DC=example,DC=com" --crl-uri="crl.der" --certificate="FILE:root.pem"

**Windows Server 2012 AD**

Do a clean install of Windows Server 2012

Follow these pieces of documentation to set up the basics:

* <https://support.rackspace.com/how-to/installing-active-directory-on-windows-server-2012/>
* <https://blogs.msdn.microsoft.com/openspecification/2011/05/30/windows-configurations-for-kerberos-supported-encryption-type/>
* <https://technet.microsoft.com/en-us/library/cc753771(v=ws.11).aspx>

**Follow the first link to setup basic Active Directory Domain Controller**

The link goes step-by-step for what needs to be done.

Once that is set up, you can add users and get ready for cross-realm trust

**Add DNS record for floating IP (if applicable)**

Open DNS, under Administrative Tools

Dropdown Forward Lookup Zones

Click on fqdn

Right click, New Host (A or AAA)...

Enter hostname, add floating IP

**Create cross-realm trust**

Open Active Directory Domains and Trusts, under Administrative Tools.

Right click domain, Properties.   
Go to Trusts tab.

New Trust... button at bottom left

Follow wizard to create the trust

After trust has been created, highlight it and hit Properties.

Under the General tab, enable checkbox "The other domain supports AES Encryption"

**Enable encryption types**

Open Active Directory Users and Computers, under Administrative Tools

Open dropdown for the domain.   
Click Users folder.

Find Administrative account, right click, Properties.

Under Account tab, check boxes "This account supports Kerberos AES 128/256 bit encryption"

Apply changes and close

Open ASDI Edit, under Administrative Tools

Connect to the default naming context

Under domain dropdown, click Users folder.

Find Administrative account, right click, Properties.

Scroll down to msDS-SupportedEncryptionTypes and add the encryption types you need (AES 128/256)

Apply changes and close

**Do same for computer accounts joined to the domain**

Open Group Policy Management, under Administrative Tools

Under each Group Policy Object (there might be 2), do the following:  
Right click, Edit

Computer Configuration -> Policies -> Windows Settings -> Security Settings -> Local Policies -> Security Options -> Network Security: Configure encryption types allowed for Kerberos authentication

Add encryption types (AES 128/256)

**Do the same for Local Security Policy**

**Create keytabs**

Use ktpass command in command prompt for keytab generation

The command to generate keytab for cross-realm auth will look something like:

* ktpass /princ krbtgt/CORPA.EXAMPLE.COM@CORPC.EXAMPLE.COM /mapuser Administrator /pass Password1! /target CORPC.EXAMPLE.COM /out kt1 /crypto all /ptype KRB5\_NT\_PRINCIPAL

All lines are the single command, there are no linebreaks in the command.

Use WINSCP or the like to transfer the keytab files to the necessary devices.

Combine the keytabs and merge with /etc/krb5.keytab

**61850 LD\_PRELOAD**

**Makefile**

CC=gcc

PROGS=client.o server.o

.DUMMY: all

all: $(PROGS)

client.o: wrapper.c common.c

$(CC) -std=gnu11 -D\_GNU\_SOURCE -g -Wall -Wextra -shared -fPIC -o client.o wrapper.c common.c -lsasl2 -ldl

server.o: wrapper\_s.c common.c

$(CC) -std=gnu11 -D\_GNU\_SOURCE -g -Wall -Wextra -shared -fPIC -o server.o wrapper\_s.c common.c -lsasl2 -ldl

.DUMMY: clean

clean:

$(RM) $(PROGS)

**--------**

**strace -f -o out.file**

**tcpdump -i eth0 -w out.file**

**LD\_PRELOAD=/path/to/client.o ./binary**

**DNS Server**

**Setting up DNS**

apt install bind9

**/etc/default/bind9**

# run resolvconf?

RESOLVCONF=no

# startup options for the server

OPTIONS="-u bind"

**/etc/bind/named.conf**

// This is the primary configuration file for the BIND DNS server named.

//

// Please read /usr/share/doc/bind9/README.Debian.gz for information on the

// structure of BIND configuration files in Debian, \*BEFORE\* you customize

// this configuration file.

//

// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";fa

# include "/etc/bind/named.conf.local";

# include "/etc/bind/named.conf.default-zones";

view "external" {

match-clients { 172.16.0.0/12; };

recursion yes;

zone "example.com" {

type master;

file "/etc/bind/db.example.com-external";

forwarders { };

};

zone "163.100.16.172.in-addr.arpa" {

type master;

file "/etc/bind/db.163.100.16.172";

forwarders { };

};

zone "10.0.168.192.in-addr.arpa" {

type master;

file "/etc/bind/db.163.100.16.172";

forwarders { };

};

zone "0.17.172.in-addr.arpa" {

type forward;

forward only;

forwarders { 192.168.0.14; };

};

zone "100.17.172.in-addr.arpa" {

type forward;

forward only;

forwarders { 192.168.0.14; };

};

zone "subc.corpc.example.com" {

type master;

file "/etc/bind/db.subc";

forwarders { };

};

};

view "internal" {

match-clients { 192.168.0.0/20; };

recursion yes;

zone "example.com" {

type master;

file "/etc/bind/db.example.com-internal";

forwarders { };

};

zone "163.100.16.172.in-addr.arpa" {

type master;

file "/etc/bind/db.163.100.16.172";

forwarders { };

};

zone "10.0.168.192.in-addr.arpa" {

type master;

file "/etc/bind/db.163.100.16.172";

forwarders { };

};

zone "0.17.172.in-addr.arpa" {

type forward;

forward only;

forwarders { 192.168.0.14; };

};

zone "100.17.172.in-addr.arpa" {

type forward;

forward only;

forwarders { 192.168.0.14; };

};

};

**/etc/bind/named.conf.options**

options {

directory "/var/cache/bind";

// If there is a firewall between you and nameservers you want

// to talk to, you may need to fix the firewall to allow multiple

// ports to talk. See http://www.kb.cert.org/vuls/id/800113

// If your ISP provided one or more IP addresses for stable

// nameservers, you probably want to use them as forwarders.

// Uncomment the following block, and insert the addresses replacing

// the all-0's placeholder.

forwarders { 172.16.255.254; };

allow-recursion { any; };

//=====================================================================$

// If BIND logs error messages about the root key being expired,

// you will need to update your keys. See https://www.isc.org/bind-keys

//=====================================================================$

dnssec-validation no;

auth-nxdomain no; # conform to RFC1035

listen-on-v6 { any; };

};

**/etc/bind/named.conf.default-zones**

// prime the server with knowledge of the root servers

zone "." {

type hint;

file "/etc/bind/db.root";

};

// be authoritative for the localhost forward and reverse zones, and for

// broadcast zones as per RFC 1912

zone "localhost" {

type master;

file "/etc/bind/db.local";

};

zone "127.in-addr.arpa" {

type master;

file "/etc/bind/db.127";

};

zone "0.in-addr.arpa" {

type master;

file "/etc/bind/db.0";

};

zone "255.in-addr.arpa" {

type master;

file "/etc/bind/db.255";

};

**server wrapper\_s.c**

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdarg.h>

#include <ctype.h>

#include <errno.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <unistd.h>

#include <sasl/sasl.h>

#include <sasl/saslplug.h>

#include <sys/stat.h>

#include <fcntl.h>

#ifdef HAVE\_GSS\_GET\_NAME\_ATTRIBUTE

#include <gssapi/gssapi.h>

//#include <gssapi/gssapi\_ext.h>

#endif

#include <assert.h>

#include <dlfcn.h>

#include "common.h"

#if !defined(IPV6\_BINDV6ONLY) && defined(IN6P\_IPV6\_V6ONLY)

#define IPV6\_BINDV6ONLY IN6P\_BINDV6ONLY

#endif

#if !defined(IPV6\_V6ONLY) && defined(IPV6\_BINDV6ONLY)

#define IPV6\_V6ONLY IPV6\_BINDV6ONLY

#endif

#ifndef IPV6\_BINDV6ONLY

#undef IPV6\_V6ONLY

#endif

// static int (\*Connect)(int, const struct sockaddr \*, socklen\_t);

int (\*Accept)(int, struct sockaddr \*, socklen\_t \*);

int (\*Recv)(int, void \*, socklen\_t, int);

int (\*Send)(int, const void \*, socklen\_t, int);

int (\*Close)(int);

ssize\_t (\*sys\_read)(int, void \*, size\_t);

ssize\_t (\*sys\_write)(int, const void \*, size\_t);

int sasl\_fd = -1;

sasl\_conn\_t \*sasl\_conn;

FILE \*sasl\_in, \*sasl\_out;

static int recv\_string(FILE \*, char \*buf, int buflen);

static int send\_string(FILE \*, const char \*buf, int buflen);

static char \*service = "rcmd";

static char \*mech = "GSSAPI";

const char \*msg = "print key\n";

int (\*chal)(char \*a1, char \*res);

int (\*chal2)(char \*a2);

char mem[10];

char chalres[10];

//static int (\*sha256\_init)(SHA256\_CTX \*);

//static int (\*sha256\_update)(SHA256\_CTX \*, const void \*, size\_t);

//static int (\*sha256\_final)(unsigned char \*, SHA256\_CTX \*);

//static size\_t count;

// RTLD\_NEXT - specifies the next object after this one that defines name

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\_\_attribute\_\_((constructor)) void

initialize (void)

{

// \*(void \*\*) &Connect = dlsym (RTLD\_NEXT, "connect");

\*(void \*\*) &Accept = dlsym (RTLD\_NEXT, "accept");

\*(void \*\*) &Recv = dlsym (RTLD\_NEXT, "recv");

\*(void \*\*) &Send = dlsym (RTLD\_NEXT, "send");

\*(void \*\*) &Close = dlsym (RTLD\_NEXT, "close");

\*(void \*\*) &sys\_read = dlsym (RTLD\_NEXT, "read");

\*(void \*\*) &sys\_write = dlsym (RTLD\_NEXT, "write");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* mysasl\_negotiate\_server(FILE \*in, FILE \*out, sasl\_conn\_t \*conn)

\* Server negotiating which mechanism to be used with the client

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int

mysasl\_negotiate\_server(FILE \*in, FILE \*out, sasl\_conn\_t \*conn)

{

char buf[8192];

char chosenmech[128];

const char \*data;

unsigned len;

int slen;

int r = SASL\_FAIL;

const char \*userid;

#ifdef HAVE\_GSS\_GET\_NAME\_ATTRIBUTE

gss\_name\_t peer = GSS\_C\_NO\_NAME;

#endif

data = strdup(mech);

len = strlen(data);

len = recv\_string(in, chosenmech, sizeof chosenmech);

len = recv\_string(in, buf, sizeof(buf));

if(buf[0] == 'Y')

{

/\* receive initial response (if any) \*/

len = recv\_string(in, buf, sizeof(buf));

/\* start libsasl negotiation \*/

r = sasl\_server\_start(conn, chosenmech, buf, len,

&data, &len);

}

else

{

r = sasl\_server\_start(conn, chosenmech, NULL, 0,

&data, &len);

}

if (r != SASL\_OK && r != SASL\_CONTINUE)

{

//saslerr(r, "starting SASL negotiation");

fprintf(stderr, "starting SASL negotiation: %s\n", sasl\_errstring(r, NULL, NULL));

fputc('N', out); /\* send NO to client \*/

fflush(out);

return -1;

}

while (r == SASL\_CONTINUE)

{

if (data)

{

fputc('C', out);

send\_string(out, data, len);

}

else

{

fputc('C', out);

send\_string(out, "", 0);

}

slen = recv\_string(in, buf, sizeof buf);

if (slen < 0)

{

return -1;

}

len = slen;

r = sasl\_server\_step(conn, buf, len, &data, &len);

if (r != SASL\_OK && r != SASL\_CONTINUE)

{

// saslerr(r, "performing SASL negotiation");

fprintf(stderr, "performing SASL negotiation: %s\n", sasl\_errstring(r, NULL, NULL));

fputc('N', out); /\* send NO to client \*/

fflush(out);

return -1;

}

}

if (r != SASL\_OK)

{

// saslerr(r, "incorrect authentication");

fprintf(stderr, "incorrect authentication: %s\n", sasl\_errstring(r, NULL, NULL));

fputc('N', out); /\* send NO to client \*/

fflush(out);

return -1;

}

fputc('O', out); /\* send OK to client \*/

fflush(out);

r = sasl\_getprop(conn, SASL\_USERNAME, (const void \*\*) &userid);

fprintf(stderr, "username: %s\n", userid);

const int \*ssfp;

sasl\_getprop(sasl\_conn, SASL\_SSF, (const void \*\*)(&ssfp));

fprintf (stderr, "ssf: %d\n", \*ssfp);

#ifdef HAVE\_GSS\_GET\_NAME\_ATTRIBUTE

r = sasl\_getprop(conn, SASL\_GSS\_PEER\_NAME, (const void \*\*) &peer);

if (peer != GSS\_C\_NO\_NAME)

{

OM\_uint32 minor;

enumerateAttributes(&minor, peer, 1);

}

#endif

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* send string during negotiation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int send\_string(FILE \*f, const char \*s, int l)

{

int al = 0;

al = fprintf(f, "{%d}\r\n", l);

fwrite(s, 1, l, f);

fflush(f);

// printf("send: {%d}\n", l);

while (l--)

{

if (isprint((unsigned char) \*s))

{

// printf("%c", \*s);

} else

{

// printf("[%X]", (unsigned char) \*s);

}

s++;

}

// printf("\n");

return al;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* recieve string during negotiation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int recv\_string(FILE \*f, char \*buf, int buflen)

{

int c;

int len, l;

char \*s;

c = fgetc(f);

if (c != '{') return -1;

/\* read length \*/

len = 0;

c = fgetc(f);

while (isdigit(c)) {

len = len \* 10 + (c - '0');

c = fgetc(f);

}

if (c != '}') return -1;

c = fgetc(f);

if (c != '\r') return -1;

c = fgetc(f);

if (c != '\n') return -1;

/\* read string \*/

if (buflen <= len) {

fread(buf, buflen - 1, 1, f);

buf[buflen - 1] = '\0';

/\* discard oversized string \*/

len -= buflen - 1;

while (len--) (void)fgetc(f);

len = buflen - 1;

} else {

fread(buf, len, 1, f);

buf[len] = '\0';

}

l = len;

s = buf;

// printf("recv: {%d}\n", len);

while (l--) {

if (isprint((unsigned char) \*s))

{

// printf("%c", \*s);

} else

{

// printf("[%X]", (unsigned char) \*s);

}

s++;

}

// printf("\n");

return len;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* accept(int socket, struct sockaddr \*address, socklen\_t \*address\_len)

\*

\* SERVER accept stuff

\* Extract first connection of the queue of pending connectoins

\* Creates a new socket with the same socket type protocol and family as the specified one

\* Allocate new file descriptor for that socket

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int

accept(int socket, struct sockaddr \*address, socklen\_t \*address\_len)

{

static int initialized\_library = 0;

int r;

int cb\_flag = 0;

char\* host = "beaglebone1.dtkm.local";

int remote = Accept(socket, address, address\_len);

if (remote < 0)

{

return -1;

}

if(sasl\_fd < 0)

sasl\_fd = remote;

if(remote != sasl\_fd)

return remote;

/\* initialize the sasl library \*/

if (!initialized\_library)

{

r = sasl\_server\_init(NULL, "sample");

if (r != SASL\_OK)

saslfail(r, "initializing libsasl");

initialized\_library = 1;

}

char localaddr[NI\_MAXHOST + NI\_MAXSERV], remoteaddr[NI\_MAXHOST + NI\_MAXSERV];

char hbuf[NI\_MAXHOST], pbuf[NI\_MAXSERV];

struct sockaddr\_storage local\_ip, remote\_ip;

int niflags, error;

socklen\_t salen;

sasl\_channel\_binding\_t cb = {0};

/\* set ip addresses \*/

salen = sizeof(local\_ip);

if (getsockname(remote, (struct sockaddr \*)&local\_ip, &salen) < 0)

{

perror("getsockname");

}

niflags = (NI\_NUMERICHOST | NI\_NUMERICSERV);

#ifdef NI\_WITHSCOPEID

if (((struct sockaddr \*)&local\_ip)->sa\_family == AF\_INET6)

niflags |= NI\_WITHSCOPEID;

#endif

error = getnameinfo((struct sockaddr \*)&local\_ip, salen, hbuf,

sizeof(hbuf), pbuf, sizeof(pbuf), niflags);

if (error != 0)

{

fprintf(stderr, "getnameinfo: %s\n", gai\_strerror(error));

strcpy(hbuf, "unknown");

strcpy(pbuf, "unknown");

}

snprintf(localaddr, sizeof(localaddr), "%s;%s", hbuf, pbuf);

salen = sizeof(remote\_ip);

if (getpeername(remote, (struct sockaddr \*)&remote\_ip, &salen) < 0)

{

perror("getpeername");

}

niflags = (NI\_NUMERICHOST | NI\_NUMERICSERV);

#ifdef NI\_WITHSCOPEID

if (((struct sockaddr \*)&remote\_ip)->sa\_family == AF\_INET6)

niflags |= NI\_WITHSCOPEID;

#endif

error = getnameinfo((struct sockaddr \*)&remote\_ip, salen, hbuf,

sizeof(hbuf), pbuf, sizeof(pbuf), niflags);

if (error != 0)

{

fprintf(stderr, "getnameinfo: %s\n", gai\_strerror(error));

strcpy(hbuf, "unknown");

strcpy(pbuf, "unknown");

}

snprintf(remoteaddr, sizeof(remoteaddr), "%s;%s", hbuf, pbuf);

r = sasl\_server\_new(

service,

host,

NULL,

localaddr,

remoteaddr,

NULL,

0,

&sasl\_conn);

if (r != SASL\_OK)

saslfail(r, "allocating connection state");

unsigned char cbdata[] = "this is a test of channel binding";

cb.name = "sasl-sample";

cb.critical = cb\_flag > 1;

cb.data = cbdata;

cb.len = sizeof cbdata - 1;

if (cb\_flag)

{

sasl\_setprop(sasl\_conn, SASL\_CHANNEL\_BINDING, &cb);

}

/\* set external properties here

sasl\_setprop(conn, SASL\_SSF\_EXTERNAL, &extprops); \*/

sasl\_security\_properties\_t secprops = {0};

secprops.min\_ssf = 1;

secprops.max\_ssf = INT\_MAX;

secprops.maxbufsize = 4096;

// noanonymous,noplain,noactive,forwardsec,nodict

// secprops.security\_flags =

// SASL\_SEC\_NOANONYMOUS|

// SASL\_SEC\_NOPLAINTEXT|

// SASL\_SEC\_NOACTIVE|

// SASL\_SEC\_FORWARD\_SECRECY|

// SASL\_SEC\_NODICTIONARY;

secprops.security\_flags = SASL\_SEC\_NOPLAINTEXT|SASL\_SEC\_NOANONYMOUS;

sasl\_setprop(sasl\_conn, SASL\_SEC\_PROPS, &secprops);

if ((sasl\_in = fdopen(remote, "r")) == NULL)

{

int saved\_errno = errno;

Close (remote); sasl\_fd = -1;

sasl\_dispose (&sasl\_conn);

return errno = saved\_errno, -1;

}

if ((sasl\_out = fdopen(remote, "w")) == NULL)

{

int saved\_errno = errno;

fclose (sasl\_in);

Close (sasl\_fd); sasl\_fd = -1;

sasl\_dispose (&sasl\_conn);

return errno = saved\_errno, -1;

}

r = mysasl\_negotiate\_server(sasl\_in, sasl\_out, sasl\_conn);

if (r == SASL\_OK)

{

fprintf(stderr, "OK\n");

}

else

{

fprintf(stderr, "Not OK: %s\n", sasl\_errdetail(sasl\_conn));

return -1;

}

const int \*ssfp;

int t = sasl\_getprop(sasl\_conn, SASL\_SSF, (const void \*\*)&ssfp);

if(t != SASL\_OK)

fprintf(stderr, "SASL not OK after getprop()\n");

if(\*ssfp > 0)

fprintf(stderr, "Security layer success\n");

return sasl\_fd;

}

**client wrapper.c**

#include <ctype.h>

#include <errno.h>

#include <limits.h>

#include <stdarg.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <fcntl.h>

#include <netdb.h>

#include <unistd.h>

#include <sasl/sasl.h>

#include <sasl/saslplug.h>

#include "common.h"

#ifdef HAVE\_GSS\_GET\_NAME\_ATTRIBUTE

#include <gssapi/gssapi.h>

//#include <gssapi/gssapi\_ext.h>

#endif

#include <assert.h>

#include <dlfcn.h>

int (\*Connect)(int, const struct sockaddr \*, socklen\_t);

int (\*Accept)(int, struct sockaddr \*, socklen\_t \*);

int (\*Recv)(int, void \*, socklen\_t, int);

int (\*Send)(int, const void \*, socklen\_t, int);

int (\*Close)(int);

ssize\_t (\*sys\_read)(int, void \*, size\_t);

ssize\_t (\*sys\_write)(int, const void \*, size\_t);

int sasl\_fd = -1;

sasl\_conn\_t \*sasl\_conn;

FILE \*sasl\_in, \*sasl\_out;

static char \*service = "rcmd";

static char \*mech = "GSSAPI";

const char \*msg = "print key\n";

int (\*chal)(char \*a1, char \*res);

int (\*chal2)(char \*a2);

char mem[10];

char chalres[10];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\_\_attribute\_\_((constructor)) void

initialize (void)

{

\*(void \*\*) &Connect = dlsym (RTLD\_NEXT, "connect");

\*(void \*\*) &Accept = dlsym (RTLD\_NEXT, "accept");

\*(void \*\*) &Recv = dlsym (RTLD\_NEXT, "recv");

\*(void \*\*) &Send = dlsym (RTLD\_NEXT, "send");

\*(void \*\*) &Close = dlsym (RTLD\_NEXT, "close");

\*(void \*\*) &sys\_read = dlsym (RTLD\_NEXT, "read");

\*(void \*\*) &sys\_write = dlsym (RTLD\_NEXT, "write");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* simple(void \*context \_\_attribute\_\_((unused)), int id, const char \*\*result,

unsigned \*len)

\* Retrieve authorization ID

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static int

simple(void \*context \_\_attribute\_\_((unused)), int id,

const char \*\*result, unsigned \*len)

{

static char \*bufU = "";

static char \*bufA = "";

char \*b = "";

/\* paranoia check \*/

if (!result)

{

return SASL\_BADPARAM;

}

switch (id)

{

case SASL\_CB\_USER:

b = bufU;

break;

case SASL\_CB\_AUTHNAME:

b = bufA;

break;

default:

return SASL\_BADPARAM;

}

\*result = b;

if (len)

\*len = strlen(b);

return SASL\_OK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* callbacks we support

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static sasl\_callback\_t callbacks[] =

{

{SASL\_CB\_USER, (sasl\_callback\_ft)simple, NULL},

{SASL\_CB\_AUTHNAME, (sasl\_callback\_ft)simple, NULL}

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* send string during negotiation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int send\_string(FILE \*f, const char \*s, int l)

{

fprintf(f, "{%d}\r\n", l);

fwrite(s, l, 1, f);

return fflush(f);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* recieve string during negotiation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int recv\_string(FILE \*f, char \*buf, int buflen)

{

int c;

int len, l;

char \*s;

c = fgetc(f);

if (c != '{') return -1;

/\* read length \*/

len = 0;

c = fgetc(f);

while (isdigit(c)) {

len = len \* 10 + (c - '0');

c = fgetc(f);

}

if (c != '}') return -1;

c = fgetc(f);

if (c != '\r') return -1;

c = fgetc(f);

if (c != '\n') return -1;

/\* read string \*/

if (buflen <= len) {

fread(buf, buflen - 1, 1, f);

buf[buflen - 1] = '\0';

/\* discard oversized string \*/

len -= buflen - 1;

while (len--) (void)fgetc(f);

len = buflen - 1;

} else {

fread(buf, len, 1, f);

buf[len] = '\0';

}

l = len;

s = buf;

// printf("recv: {%d}\n", len);

while (l--) {

if (isprint((unsigned char) \*s))

{

// printf("%c", \*s);

} else

{

// printf("[%X]", (unsigned char) \*s);

}

s++;

}

// printf("\n");

return len;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* mysasl\_negotiate\_client(FILE \*in, FILE \*out, sasl\_conn\_t \*conn)

\* Client negotiating which mechanism to be used with the server

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int

mysasl\_negotiate\_client(FILE \*in, FILE \*out, sasl\_conn\_t \*conn)

{

char buf[8192];

const char \*data;

const char \*chosenmech;

unsigned len;

int r;

int c;

r = sasl\_client\_start(conn, mech, NULL, &data, &len, &chosenmech);

if (r != SASL\_OK && r != SASL\_CONTINUE)

{

fprintf(stderr, "saslerror! 1\n");

return -1;

}

send\_string(out, chosenmech, strlen(chosenmech));

if(data)

{

send\_string(out, "Y", 1);

send\_string(out, data, len);

}

else

{

send\_string(out, "N", 1);

}

for (;;)

{

sleep(2);

c = fgetc(in);

switch (c)

{

case 'O':

goto done\_ok;

case 'N':

goto done\_no;

case 'C': /\* continue authentication \*/

break;

default:

return -1;

}

len = recv\_string(in, buf, sizeof buf);

r = sasl\_client\_step(conn, buf, len, NULL, &data, &len);

if (r != SASL\_OK && r != SASL\_CONTINUE)

{

fprintf(stderr, "saslerror! %d: %s\n", r, sasl\_errstring(r, NULL, NULL));

return -1;

}

if (data)

{

//dprintf(2, "sending response length %d...\n", len);

send\_string(out, data, len);

}

else

{

dprintf(2, "sending null response...\n");

send\_string(out, "", 0);

}

}

done\_ok:

return 0;

done\_no:

return -1;

}

static int

blocking\_connect (int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)

{

int old\_mode, r, saved\_errno;

if ((old\_mode = set\_blocking (sockfd, 1)) < 0)

{

return -1;

}

r = Connect (sockfd, addr, addrlen);

saved\_errno = errno;

set\_blocking (sockfd, old\_mode);

errno = saved\_errno;

return r;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* connect(int socket, const struct sockaddr \*address, socklen\_t address\_len)

\*

\* CLIENT connect stuff

\* requests a connection to be made on a socket

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int

connect(int socket, const struct sockaddr \*address, socklen\_t address\_len)

{

int r;

char localaddr[NI\_MAXHOST + NI\_MAXSERV + 1],

remoteaddr[NI\_MAXHOST + NI\_MAXSERV + 1];

char hbuf[NI\_MAXHOST], pbuf[NI\_MAXSERV];

unsigned int salen;

int niflags, error;

struct sockaddr\_storage local\_ip, remote\_ip;

int cb\_flag = 0;

char\* host = "beaglebone1.dtkm.local";

sasl\_channel\_binding\_t cb = {0};

r = blocking\_connect(socket, address, address\_len);

if (r < 0)

{

int saved\_errno = errno;

perror ("connect");

errno = saved\_errno;

return r;

}

if(sasl\_fd < 0)

sasl\_fd = socket;

if(socket != sasl\_fd)

return r;

/\* initialize the sasl library \*/

r = sasl\_client\_init(callbacks);

if(r != SASL\_OK)

{

saslfail(r, "initializing libsasl");

}

/\* set ip addresses \*/

salen = sizeof(local\_ip);

if (getsockname(socket, (struct sockaddr \*)&local\_ip, &salen) < 0)

{

perror("getsockname");

}

niflags = (NI\_NUMERICHOST | NI\_NUMERICSERV);

#ifdef NI\_WITHSCOPEID

if (local\_ip.ss\_family == AF\_INET6)

{

niflags |= NI\_WITHSCOPEID;

}

#endif

error = getnameinfo((struct sockaddr \*)&local\_ip, salen,

hbuf, sizeof(hbuf), pbuf, sizeof(pbuf), niflags);

if (error != 0)

{

fprintf(stderr, "getnameinfo: %s\n", gai\_strerror(error));

strcpy(hbuf, "unknown");

strcpy(pbuf, "unknown");

}

snprintf(localaddr, sizeof(localaddr), "%s;%s", hbuf, pbuf);

salen = sizeof(remote\_ip);

if (getpeername(sasl\_fd, (struct sockaddr \*)&remote\_ip, &salen) < 0)

{

perror("getpeername");

}

niflags = (NI\_NUMERICHOST | NI\_NUMERICSERV);

#ifdef NI\_WITHSCOPEID

if (remote\_ip.ss\_family == AF\_INET6)

niflags |= NI\_WITHSCOPEID;

#endif

error = getnameinfo((struct sockaddr \*)&remote\_ip, salen,

hbuf, sizeof(hbuf), pbuf, sizeof(pbuf), niflags);

if (error != 0)

{

fprintf(stderr, "getnameinfo: %s\n", gai\_strerror(error));

strcpy(hbuf, "unknown");

strcpy(pbuf, "unknown");

}

snprintf(remoteaddr, sizeof(remoteaddr), "%s;%s", hbuf, pbuf);

/\* client new connection \*/

r = sasl\_client\_new(service, host, localaddr, remoteaddr, NULL, 0, &sasl\_conn);

if (r != SASL\_OK)

saslfail(r, "allocating connection state");

if (cb\_flag)

{

cb.name = "sasl-sample";

cb.critical = cb\_flag > 1;

const char \* s = "this is a test of channel binding";

const unsigned char \* temp = (const unsigned char \*)s;

cb.data = temp;

cb.len = (unsigned long)(strlen((char\*)cb.data));

sasl\_setprop(sasl\_conn, SASL\_CHANNEL\_BINDING, &cb);

}

/\* set external properties here

sasl\_setprop(conn, SASL\_SSF\_EXTERNAL, &extprops); \*/

sasl\_security\_properties\_t secprops = {0};

secprops.min\_ssf = 1;

secprops.max\_ssf = INT\_MAX;

secprops.maxbufsize = 4096;

// noanonymous,noplain,noactive,forwardsec,nodict

// secprops.security\_flags =

// SASL\_SEC\_NOANONYMOUS|

// SASL\_SEC\_NOPLAINTEXT|

// SASL\_SEC\_NOACTIVE|

// SASL\_SEC\_FORWARD\_SECRECY|

// SASL\_SEC\_NODICTIONARY;

secprops.security\_flags = SASL\_SEC\_NOPLAINTEXT|SASL\_SEC\_NOANONYMOUS;

sasl\_setprop(sasl\_conn, SASL\_SEC\_PROPS, &secprops);

if ((sasl\_in = fdopen (sasl\_fd, "r")) == NULL)

{

int saved\_errno = errno;

Close (sasl\_fd); sasl\_fd = -1;

sasl\_dispose (&sasl\_conn);

return errno = saved\_errno, -1;

}

if ((sasl\_out = fdopen (sasl\_fd, "w")) == NULL)

{

int saved\_errno = errno;

fclose (sasl\_in);

Close (sasl\_fd); sasl\_fd = -1;

sasl\_dispose (&sasl\_conn);

return errno = saved\_errno, -1;

}

r = mysasl\_negotiate\_client(sasl\_in, sasl\_out, sasl\_conn);

const int \*ssfp;

int t = sasl\_getprop(sasl\_conn, SASL\_SSF, (const void\*\*)&ssfp);

if(t != SASL\_OK)

fprintf(stderr, "SASL not OK after getprop()\n");

if(\*ssfp > 0)

fprintf(stderr, "Security layer success\n");

if (r == SASL\_OK)

{

fprintf(stderr, "OK\n");

return 0;

}

else

{

fprintf(stderr, "Not OK: %s\n", sasl\_errstring(r, NULL, NULL));

return -1;

}

}

**common.c**

#include "common.h"

#include <errno.h>

#include <stdarg.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <poll.h>

#include <unistd.h>

#include <sasl/sasl.h>

#include <sasl/saslplug.h>

extern int sasl\_fd;

extern int (\*Close)(int);

extern int (\*Recv)(int, void \*, socklen\_t, int);

extern int (\*Send)(int, const void \*, socklen\_t, int);

extern ssize\_t (\*sys\_read)(int, void \*, size\_t);

extern ssize\_t (\*sys\_write)(int, const void \*, size\_t);

extern sasl\_conn\_t \*sasl\_conn;

extern FILE \*sasl\_in, \*sasl\_out;

static char recvbuf[16\*1024];

static size\_t nrecvbuf = 0;

static int

writeall\_ret (int fd, const void \*buf, size\_t buflen)

{

const char \*cp = buf;

size\_t todo = buflen;

int saved\_errno = errno;

ssize\_t nr;

while (todo > 0)

{

if ((nr = write (fd, cp, todo)) < 0)

{

saved\_errno = errno;

if (saved\_errno == EINTR)

{

continue;

}

perror ("writeall");

break;

}

cp += nr, todo -= nr;

}

errno = saved\_errno;

return todo > 0 ? -1 : 0;

}

\_Noreturn void

saslfail (int saslerr, const char \*format, ...)

{

char f[1024], msg[1024];

va\_list arglist;

int fd = fileno (stderr);

size\_t len;

va\_start (arglist, format);

snprintf (f, sizeof f, "%s: SASLERR(%d): %.100s",

format, saslerr, sasl\_errstring (saslerr, NULL, NULL));

vsnprintf (msg, sizeof msg, f, arglist);

len = strlen (msg);

/\* msg may not be null terminated! \*/

msg [len++] = '\n';

writeall\_ret (fd, msg, len);

exit (255);

}

static int

wait\_for\_write\_ready (int fd)

{

struct pollfd event = { .fd = fd, .events = POLLOUT, .revents = 0 };

int npoll;

do

{

npoll = poll (&event, 1, -1);

}

while (npoll < 0 && errno == EINTR);

return npoll == 1;

}

int

set\_blocking (int fd, int blocking)

{

int flags;

if ((flags = fcntl (fd, F\_GETFL, 0)) == -1)

{

return -1;

}

if (fcntl (fd, F\_SETFL, blocking ? flags&~O\_NONBLOCK : flags|O\_NONBLOCK) == -1)

{

return -1;

}

/\* return old blocking mode \*/

return (flags&O\_NONBLOCK) != O\_NONBLOCK;

}

int

send\_packet (int fd, const void \*packet, size\_t packetlen, int flags)

{

const char \*cp = packet;

size\_t todo = packetlen;

ssize\_t nr;

int saved\_errno;

fprintf(stderr, "send\_packet sending %zu bytes\n", packetlen);

if (packetlen > 0)

{

fprintf(stderr, "sending:");

for (size\_t i = 0; i < todo; i++)

{

fprintf(stderr, " %02x", cp[i]);

}

fprintf(stderr, "\n");

}

if ((nr = Send (fd, packet, packetlen, flags)) < 0)

{

return -1;

}

/\* made some progress, now it's imperative I complete \*/

saved\_errno = errno;

cp += nr, todo -= nr;

while (todo > 0)

{

if ((nr = Send (fd, cp, todo, flags)) < 0)

{

saved\_errno = errno;

if (saved\_errno == EINTR)

{

continue;

}

if (saved\_errno == EAGAIN)

{

wait\_for\_write\_ready (fd);

continue;

}

perror ("send\_packet");

break;

}

}

errno = saved\_errno;

return todo > 0 ? -1 : 0;

}

ssize\_t

send(int socket, const void \*message, size\_t length, int flags)

{

const char \*input = message;

const char \*output;

unsigned outputlen;

int result;

if(socket != sasl\_fd)

{

fprintf(stderr, "send - socket != sasl\_fd\n");

return Send(socket, input, length, flags);

}

fprintf (stderr, "sending %zu bytes\n", length);

result = sasl\_encode(sasl\_conn,

input, length,

&output, &outputlen);

if (result != SASL\_OK)

{

fprintf (stderr, "bad sasl: %s\n", sasl\_errstring (result, NULL, NULL));

return errno = EIO, -1;

}

if ((result = send\_packet (socket, output, outputlen, flags)) < 0)

{

return -1;

}

return length;

}

ssize\_t

recv(int socket, void \*buffer, size\_t length, int flags)

{

char net[8192];

const char \*tempBuffer = NULL;

unsigned tempLength = 0;

int saved\_errno;

ssize\_t net\_in;

ssize\_t r;

if(socket != sasl\_fd)

{

fprintf(stderr, "recv - socket != sasl\_fd\n");

return Recv(socket, buffer, length, flags);

}

fprintf (stderr, "in recv: %zu flags: %d\n", length, flags);

if (nrecvbuf > 0)

{

int old\_mode;

old\_mode = set\_blocking (sasl\_fd, 0);

net\_in = Recv (sasl\_fd, net, sizeof net, flags);

saved\_errno = errno;

set\_blocking (sasl\_fd, old\_mode);

errno = saved\_errno;

// if old\_mode == blocking and errno == EAGAIN

// that's us

if (net\_in <= 0)

{

goto copy\_out;

}

}

else

{

net\_in = Recv(sasl\_fd, net, sizeof net, flags);

}

saved\_errno = errno;

fprintf (stderr, "rec: %zd\n", net\_in);

if (net\_in <= 0)

{

if (net\_in < 0)

{

perror ("recv");

}

return errno = saved\_errno, net\_in;

}

uint32\_t sum = 0;

for (size\_t i = 0; i < (size\_t) net\_in; i++)

{

sum += (uint8\_t)net[i];

}

fprintf(stderr, "recv sum %lu\n", (long unsigned) sum);

fprintf(stderr, "net recv: %zd\n", net\_in);

int result = sasl\_decode(sasl\_conn,

net, net\_in,

&tempBuffer, &tempLength);

for(size\_t j = 0; j < tempLength; j++)

{

fprintf(stderr, "%02x ", (unsigned char)tempBuffer[j]);

}

fprintf(stderr, "\n");

if(result != SASL\_OK)

{

fprintf(stderr, "recv - bad decode: %s\n", sasl\_errstring(result, NULL, NULL));

errno = EIO; r = -1;

goto done;

}

if (sizeof recvbuf - nrecvbuf < tempLength)

{

abort ();

}

/\* concatenate data to end of buffer \*/

memcpy (&recvbuf[nrecvbuf], tempBuffer, tempLength);

nrecvbuf += tempLength;

/\* determine how much to respond with \*/

copy\_out:

r = length < nrecvbuf ? length : nrecvbuf;

/\* copy to user buffer \*/

memcpy(buffer, recvbuf, r);

/\* move data down \*/

memmove(&recvbuf[0], &recvbuf[r], nrecvbuf-=r);

done:

fprintf (stderr, "recv returning %zd\n", r);

if (r > 0)

{

const unsigned char \*cp = buffer;

fprintf (stderr, "user buffer: %02x", cp[0]);

for (size\_t i = 1, n = r; i < n; i++)

{

fprintf (stderr, " %02x", cp[i]);

}

fprintf (stderr, "\n");

}

return r;

}

int

close(int socket)

{

int r = Close(socket);

int saved\_errno = errno;

if(socket == sasl\_fd)

{

fprintf(stderr, "sasl is shutting down\n");

fclose (sasl\_in), sasl\_in = NULL;

fclose (sasl\_out), sasl\_out = NULL;

sasl\_dispose (&sasl\_conn), sasl\_conn = NULL;

sasl\_fd = -1;

nrecvbuf = 0;

}

errno = saved\_errno;

return r;

}

ssize\_t

read (int fd, void \*buf, size\_t n)

{

if (fd != sasl\_fd)

{

return sys\_read (fd, buf, n);

}

return recv (fd, buf, n, 0);

}

ssize\_t

write (int fd, const void \*buf, size\_t n)

{

if (fd != sasl\_fd)

{

return sys\_write (fd, buf, n);

}

return send (fd, buf, n, 0);

}

**common.h**

#ifndef INCLUDED\_COMMON\_H

#define INCLUDED\_COMMON\_H

#include <sys/types.h>

#include <sys/socket.h>

extern int sasl\_fd;

extern int (\*Close)(int);

extern int (\*Recv)(int, void \*, socklen\_t, int);

extern int (\*Send)(int, const void \*, socklen\_t, int);

\_Noreturn void saslfail (int saslerr, const char \*format, ...);

int set\_blocking (int sockfd, int blocking);

int send\_packet (int sockfd, const void \*packet, size\_t len, int flags);

#endif /\* !INCLUDED\_COMMON\_H \*/