

Angels (Open SSL) and D(a)emons

PALLABI GHOSH

(PALLABIG@ANDREW.CMU.EDU)

15-441 COMPUTER NETWORKS

RECITATION 4

A solid orange horizontal bar at the bottom of the slide.

Project 1 Final Submission

(1) SSL

(2) CGI

(3) Daemonize

Extras



ssl_example.c

ssl_client.py

daemonize.c

(on course website)

SSL

Getting a...

Domain Name

Create a Domain Name

.Get a **free** domain name from **No-IP**

No-IP Free

No-IP Free is our entry level service. Use yourname.no-ip.org instead of a hard to remember IP address or URL. With No-IP Dynamic DNS, our free Dynamic Update Client keeps track of your changing IP address and updates your hostname, keeping your connection active.

[Sign Up Now](#) [More](#)

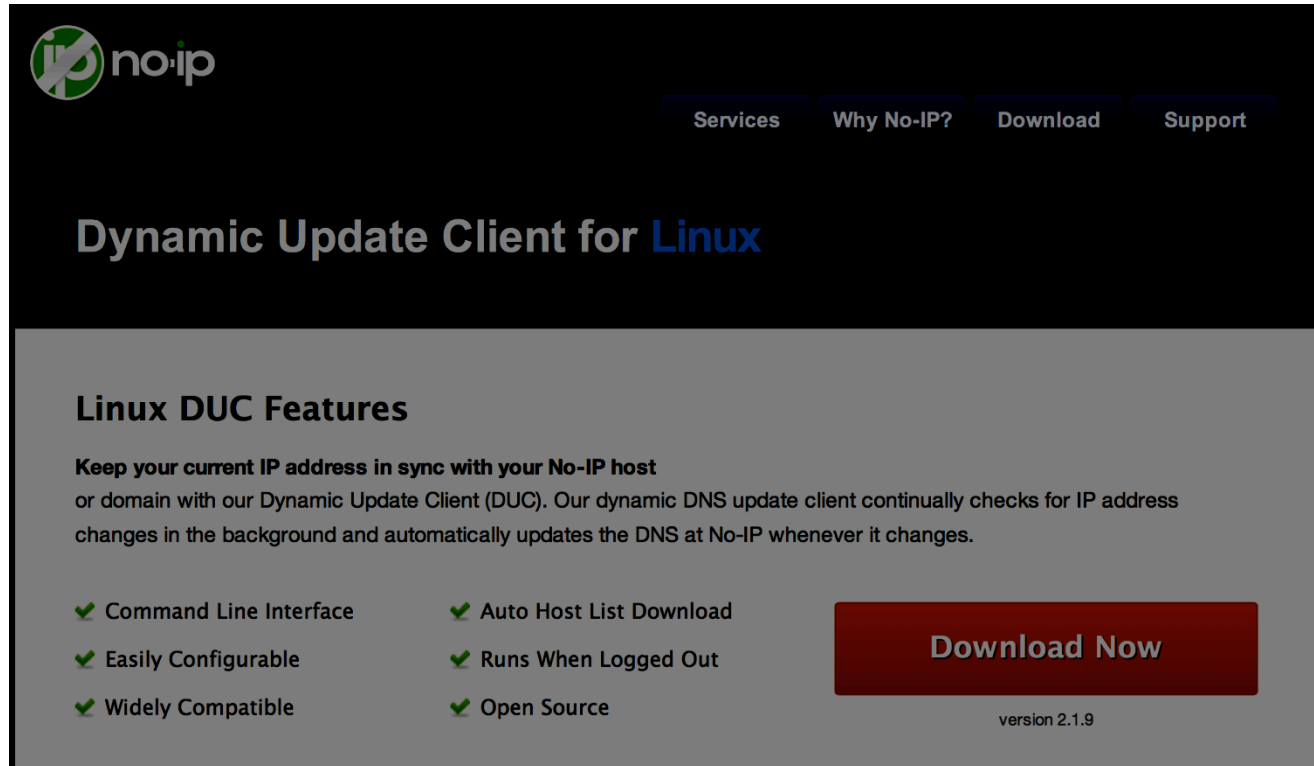


.Use your **Andrew ID** as the **hostname**

Hostname:

▼

Get the Update Client



The screenshot shows the No-IP website's page for the Dynamic Update Client (DUC) for Linux. At the top left is the No-IP logo. To its right are navigation links: Services, Why No-IP?, Download, and Support. The main heading is "Dynamic Update Client for Linux". Below this, the section "Linux DUC Features" is introduced. A paragraph explains that the DUC keeps the current IP address in sync with the No-IP host or domain by continually checking for IP address changes in the background and automatically updating the DNS at No-IP whenever it changes. Below the paragraph is a list of features, each preceded by a green checkmark icon. The features are: Command Line Interface, Auto Host List Download, Easily Configurable, Runs When Logged Out, Widely Compatible, and Open Source. To the right of the list is a large red button labeled "Download Now". Below the button, the text "version 2.1.9" is displayed.

no-ip

Services Why No-IP? Download Support

Dynamic Update Client for Linux

Linux DUC Features

Keep your current IP address in sync with your No-IP host
or domain with our Dynamic Update Client (DUC). Our dynamic DNS update client continually checks for IP address changes in the background and automatically updates the DNS at No-IP whenever it changes.

- ✓ Command Line Interface
- ✓ Auto Host List Download
- ✓ Easily Configurable
- ✓ Runs When Logged Out
- ✓ Widely Compatible
- ✓ Open Source

Download Now

version 2.1.9

.You don't have root, so...

- Just build (**make**), don't install (**make install**)
- Run manually when your IP changes

Create No-IP Conf File

./noip2 -C -c noip.conf

```
[dnaylor@unix3 ~/noip-2.1.9-1]$ ./noip2 -C -c noip.conf
```

Auto configuration for Linux client of no-ip.com.

Please enter the **login/email** string for no-ip.com **<username>**

Please enter the **password** for user '<username>' *****

Only one host [dnaylor.no-ip.biz] is registered to this account.

It will be used.

Please enter an update interval:[30]

Do you wish to run something at successful update?[N] (y/N)

New configuration file 'noip.conf' created.

Update Your IP Address

`./noip2 -c noip.conf -i 108.17.82.243`

```
[dnaylor@unix3 ~/noip-2.1.9-1]$ ./noip2 -c noip.conf -i 108.17.82.243
```

IP address detected on command line.

Running in single use mode.

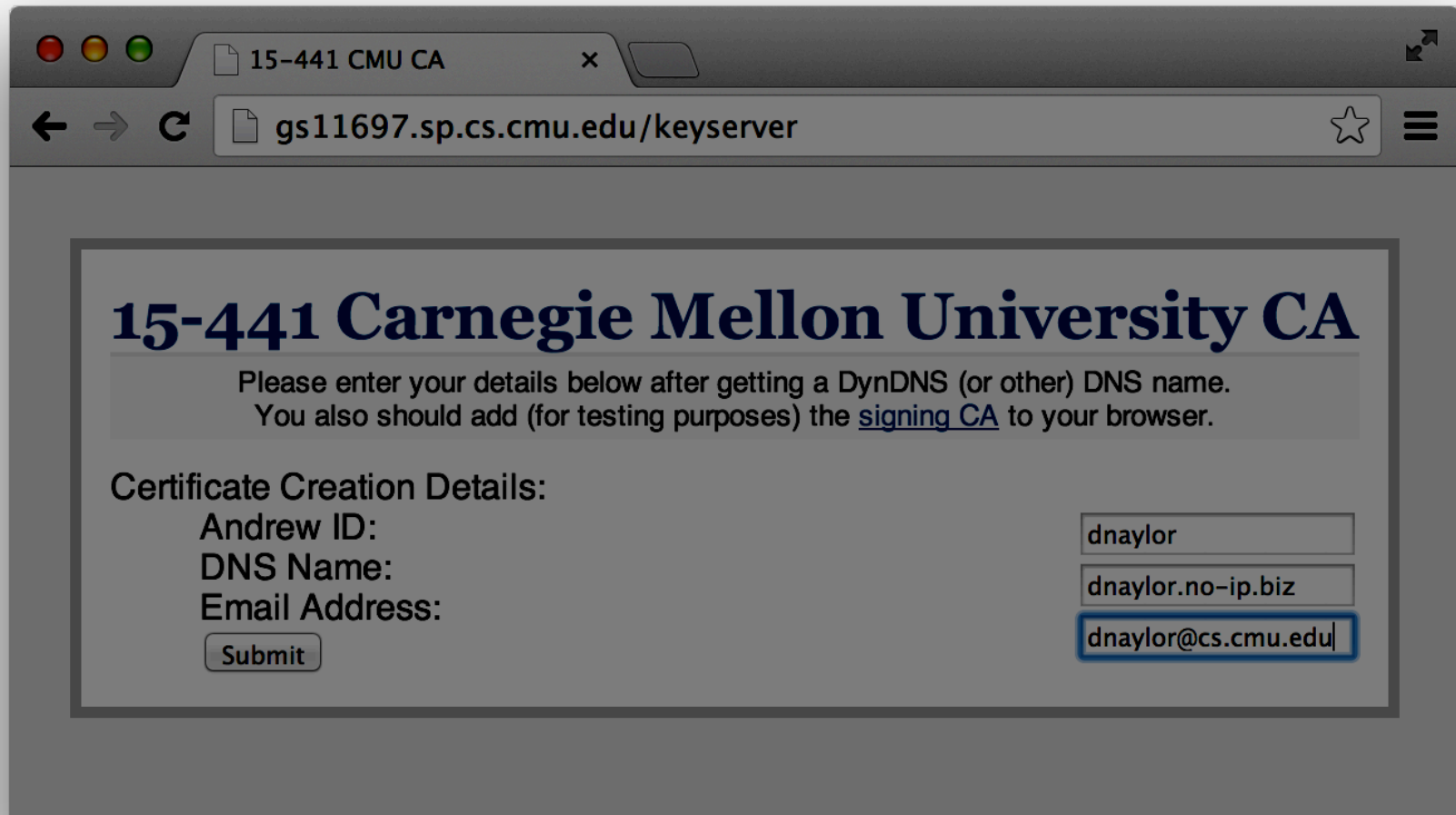
Getting a...

Certificate

15-441 Certificate Authority

Example

<http://gs11697.sp.cs.cmu.edu/keyserver>



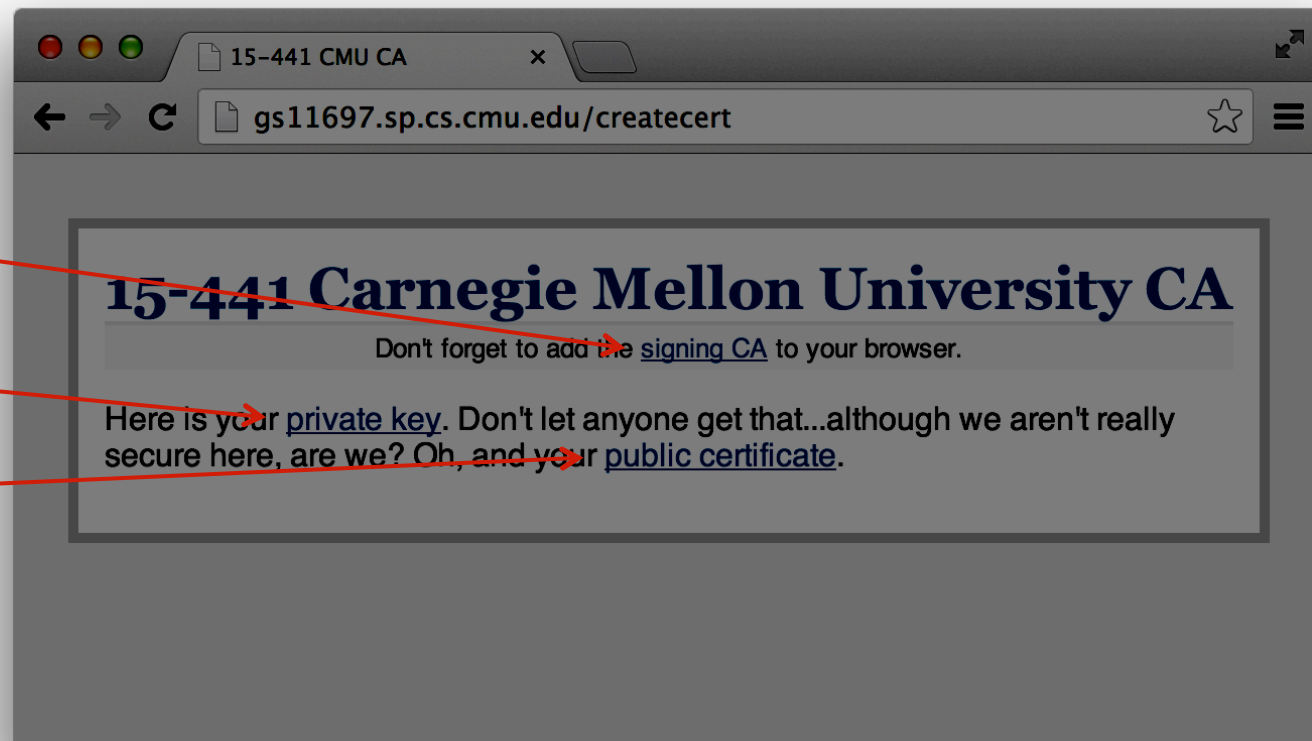
The screenshot shows a web browser window with the address bar displaying `gs11697.sp.cs.cmu.edu/keyserver`. The page title is "15-441 CMU CA". The main content area has a heading "15-441 Carnegie Mellon University CA" and a message: "Please enter your details below after getting a DynDNS (or other) DNS name. You also should add (for testing purposes) the [signing CA](#) to your browser."

Below the message, there is a section titled "Certificate Creation Details:" with three input fields and a "Submit" button:

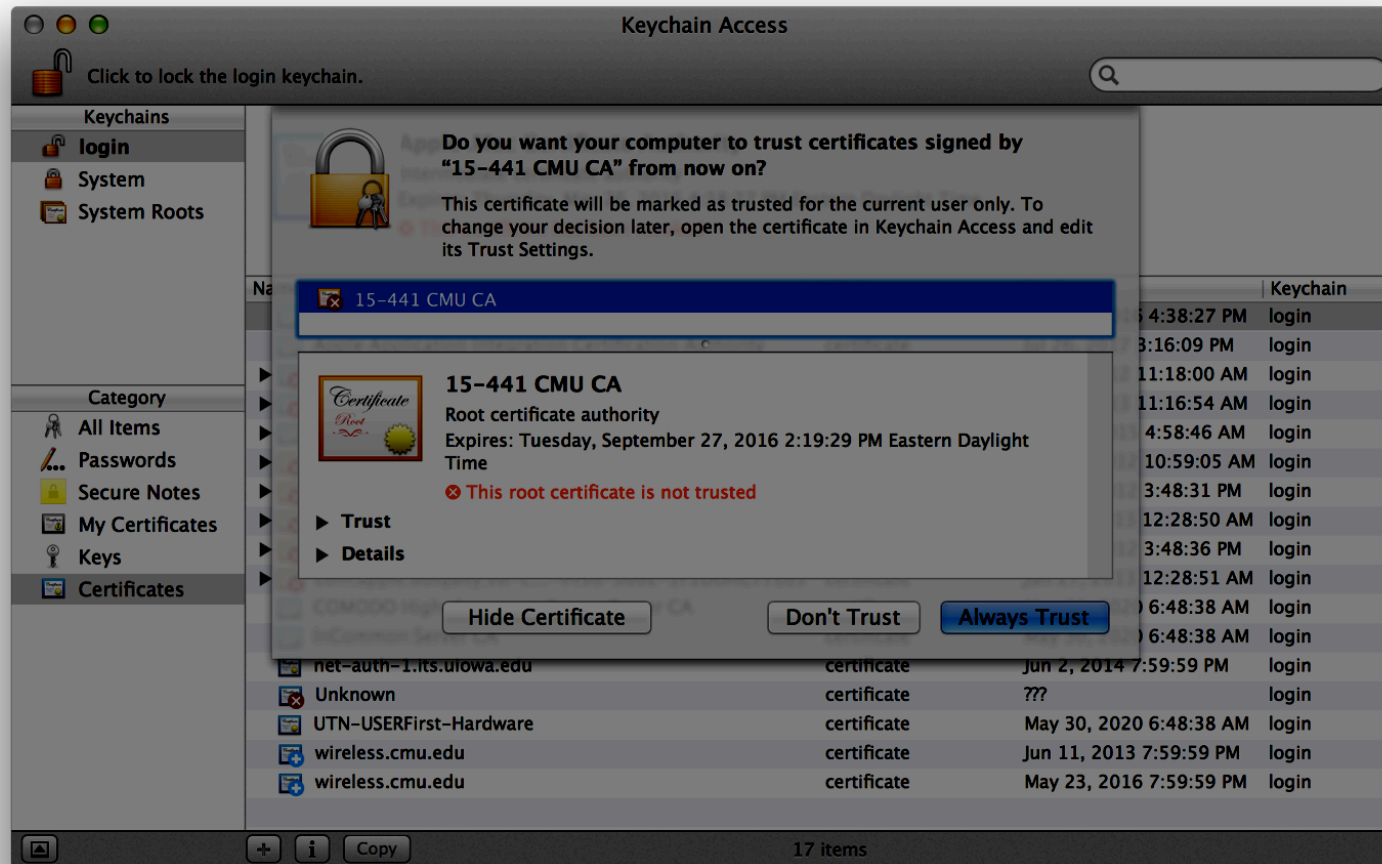
Andrew ID:	<input type="text" value="dnaylor"/>
DNS Name:	<input type="text" value="dnaylor.no-ip.biz"/>
Email Address:	<input type="text" value="dnaylor@cs.cmu.edu"/>
<input type="button" value="Submit"/>	

You Need 3 Things

- 1) CA certificate
- 2) Your private key
- 3) Your certificate



Add CA Cert to Your System/ Browser



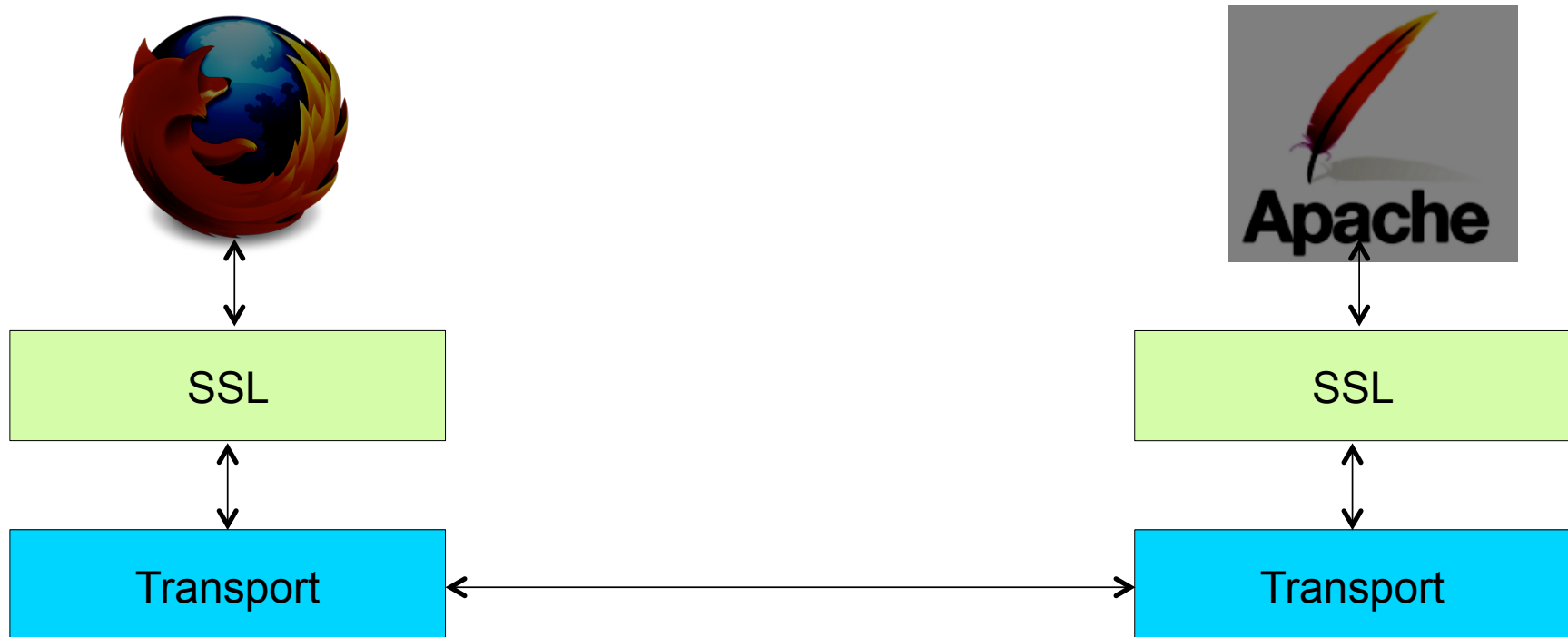
e.g., add to OSX Keychain

Implementing an...

SSL Server

What is SSL?

- Standard behind secure communication on the Internet.
- Provides confidentiality & integrity
- Sits between transport & application



OpenSSL Toolkit

- Command line tools, **SSL library**, and crypto library
- Can do a lot more than SSL
 - Message digests
 - Encryption and decryption of files
 - Digital certificates
 - Digital signatures
 - Random number generation

SSL Server In a Nutshell

- Use the **OpenSSL library**, here is a link to their [documentation](#).
- Create a **second server socket** in addition to the first one, use the passed in SSL port from the command line arguments.
- Add this socket to the **select() loop** just like your normal HTTP server socket.
- Whenever you accept connections, wrap them with the **SSL wrapping functions**.
- Use the special **read() and write() SSL functions** to read and write to these special connected clients
- In the select() loop, you need to know if a socket you are dealing with is SSL wrapped or not
- Use appropriate IO depending on the 'type' of socket---although use select() for all fd's
- Use your private key and certificate file that you obtained earlier.

Open SSL headers

```
/* OpenSSL headers */  
#include <openssl/bio.h>  
#include <openssl/ssl.h>  
#include <openssl/err.h>
```

Initialization Steps

.Global System Initialize

- `SSL_library_init()`
- `SSL_load_error_strings()`

.Initialize SSL_METHOD and SSL_CTX

- `meth=SSLv23_method();`
- `ctx=SSL_CTX_new(meth);`

.Loading keys

- `SSL_CTX_use_certificate_file(...)`
- `SSL_CTX_use_PrivateKey_file(...)`

Global Initialization

`.SSL_library_init()`

- registers the available SSL/TLS ciphers and digests.

`.SSL_load_error_strings()`

- Provide readable error messages.

SSL_METHOD

- .To describe protocol versions
- .SSLv1, SSLv2 and TLSv1

```
SSL_METHOD* meth = TLSv1_method();
```

SSL_CTX

- Data structure to store keying material
- Reused for all connections; make ONE for your server

```
SSL_CTX* ctx = SSL_CTX_new(meth);
```

SSL_CTX_use_certificate_file()

- Loads the first certificate stored in file into ctx.

- The formatting type of the certificate must be specified from the known types

 - SSL_FILETYPE_PEM

 - SSL_FILETYPE_ASN1.

 - Our CA generates files of PEM format

- `int SSL_CTX_use_certificate_file(SSL_CTX *ctx, const char *file, int type);`

SSL_CTX_use_PrivateKey_file()

.Adds the first private key found in file to ctx.

.The formatting type of the certificate must be specified from the known types:

- SSL_FILETYPE_PEM
- SSL_FILETYPE_ASN1.
- Our CA generates files of PEM format

```
.int SSL_CTX_use_PrivateKey_file(SSL_CTX *ctx, const char *file, int type);
```


Wrapping Connections

- .Create new SSL structure using `SSL_new()`
- .Connect it to the socket using `SSL_set_fd()`
- .Perform handshake using `SSL_accept()`
- .Read and write using `SSL_read()` and `SSL_write()`
- .Perform shutdown at the end, also need to clear state and close underlying I/O socket etc.
- .As always, check for return value and handle errors appropriately!

SSL_new()

- .Creates a new SSL structure
- .Create one **per connection**
- .Inherits the settings of the underlying context.

```
SSL* ssl = SSL_new(ctx);
```

SSL_set_fd()

.Tell the SSL object which socket it will wrap

```
int SSL_set_fd(SSL *ssl, int fd);
```

SSL_accept

- SSL_accept - wait for a TLS/SSL client to initiate a TLS/SSL handshake

```
int SSL_accept(SSL *ssl)
```

- (Do this after a standard **accept()**.)

SSL_read and SSL_write

.SSL_read to read bytes from a TLS/SSL connection

- `int SSL_read(SSL *ssl, void *buf, int num);`

.SSL_write to write bytes to a TLS/SSL connection

- `int SSL_write(SSL *ssl, const void *buf, int num);`

.NOTE:

- The data are received in records (with a maximum record size of 16kB for SSLv3/TLSv1).
- Only when a record has been completely received, it can be processed (decryption and integrity check)

SSL_shutdown

- Shuts down an active TLS/SSL connection.

- `int SSL_shutdown(SSL *ssl);`

- (Then do a standard `close()`.)

BIO - Optional

- .I/O abstraction provided by OpenSSL
- .Hides the underlying I/O and can set up connection with any I/O (socket, buffer, ssl etc)
- .BIOs can be stacked on top of each other using push and pop!
- .NOTE: You **don't** have to necessarily use BIO for this project! The next few slides describe creating BIO and working with it.

BIO_new()

- Returns a new BIO using method type.
- Check BIO_s_socket(), BIO_f_buffer(), BIO_f_ssl()
- Check BIO_new_socket()
- BIO * BIO_new(BIO_s_socket());
BIO_set_fd(sbio, sock, BIO_NOCLOSE);

SSL_set_bio()

.Connects the BIOs rbio and wbio for the read and write operations of the TLS/SSL (encrypted) side of ssl

```
void SSL_set_bio(SSL *ssl, BIO *rbio, BIO *wbio)
```

Example of Stacking BIOs

```
buf_io = BIO_new(BIO_f_buffer());
```

```
/* create a buffer BIO */
```

```
ssl_bio = BIO_new(BIO_f_ssl());
```

```
/* create an ssl BIO */
```

```
BIO_set_ssl(ssl_bio, ssl, BIO_CLOSE);
```

```
/* assign the ssl BIO to SSL */
```

```
BIO_push(buf_io, ssl_bio);
```

BIO_read() and BIO_write()

- Attempts to read len bytes from BIO b and places the data in buf.

- `int BIO_read(BIO *b, void *buf, int len);`

- Attempts to write len bytes from buf to BIO b.

- `int BIO_write(BIO *b, const void *buf, int len);`

SSL

Questions?

Daemonizing

Orphaning

- Fork the process to create a copy (child)
- Let parent exit!
- The child will become child of init process
 - Start operating in the background

```
int pid = fork();  
if (pid < 0) exit(EXIT_FAILURE); /* fork error */  
if (pid > 0) exit(EXIT_SUCCESS); /* parent exits */  
/* child (daemon) continues */
```

Process Independence

- .Process inherits parent's controlling tty; need to detach
- .Server should not receive signals from the process that started it
- .Operate independently from other processes
- .setsid() /*obtain a new process group*/

Close File Descriptors

•Close all open descriptors inherited

```
int i;  
for (i = getdtablesize(); i >= 0; --i)  
    close(i);
```

•Connect standard I/O descriptors (stdin 0, stdout 1, stderr 2) to /dev/null

```
.i = open("/dev/null",O_RDWR); /* open stdin */  
.dup(i) /* stdout */  
.dup(i) /* stderr */
```


File Creation Mask

- .Servers run as super-user
- .Need to protect the files they create
- .File creation mode is 750 (complement of 027)

```
umask(027);
```

Running Directory

.Server should run in a known directory

```
chdir("/servers/");
```

Mutual Exclusion

.We want only one copy of the server (file locking)

.Record pid of the running instance!

- 'cat lisod.lock' more efficient than 'ps -ef | grep lisod'

.

```
lfp = open(lock_file, O_RDWR|O_CREAT, 0640);
```

```
if (lfp < 0)
```

```
    exit(EXIT_FAILURE); /* cannot open */
```

```
if (lockf(lfp, F_TLOCK, 0) < 0)
```

```
    exit(EXIT_SUCCESS); /* cannot lock */
```

```
sprintf(str, "%d\n", getpid());
```

```
write(lfp, str, strlen(str)); /*record pid to lockfile */
```

Logging

.You sent stdout and stderr to /dev/null, so you need to log to a file!

Daemonizing

Questions?