# Angels (Open SSL) and D(a)emons

15-441: Computer Networks

M.Tarek Abdellatif

Fork me on GitHub

# Extras

ssl_example.c

ssl_client.py

daemonize.c

(on course website)

## PJ1 Final Submission

(1) SSL
(2) CGI
(3) Daemonize

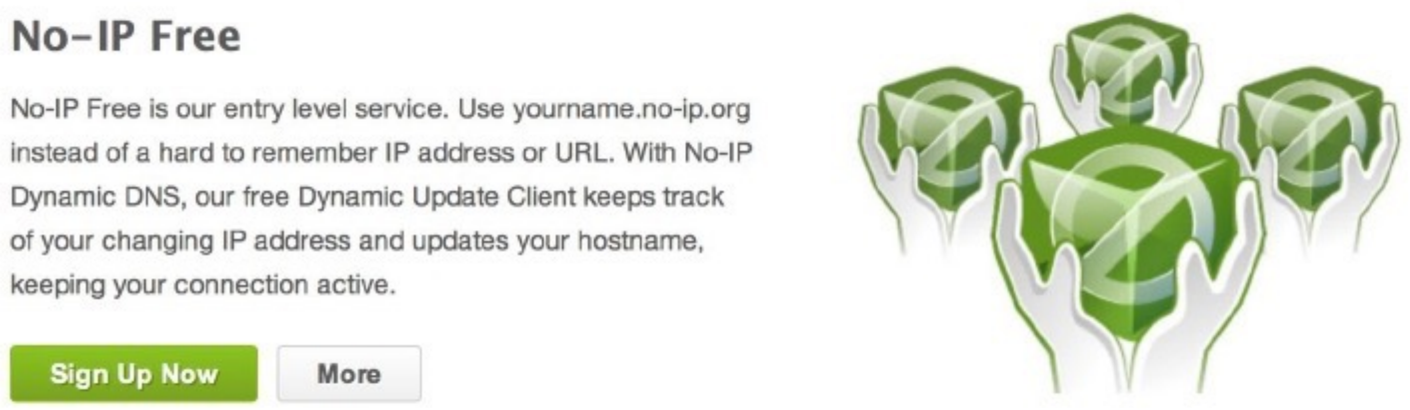
# SSL

Getting a...

## Domain Name

## Create a Domain Name

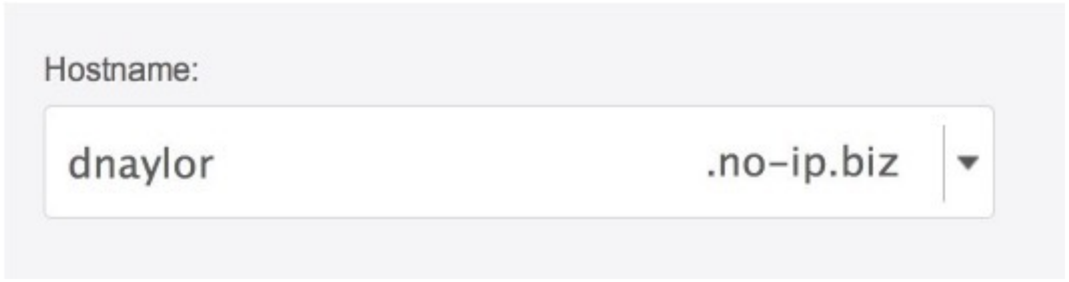
- Get a free domain name from No-IP



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.

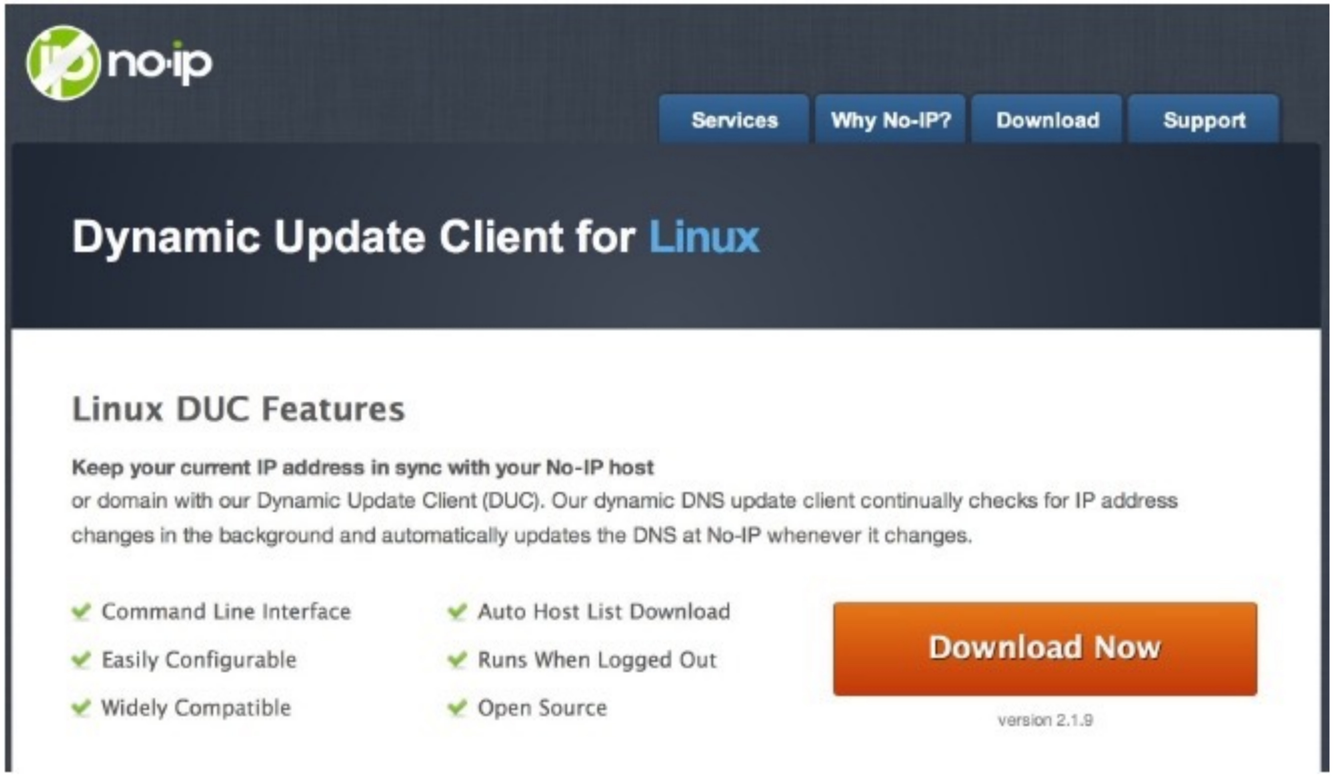
- Use your Andrew ID as the hostname



Hostname:

dnaylor .no-ip.biz

# Get the Update Client



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

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
```

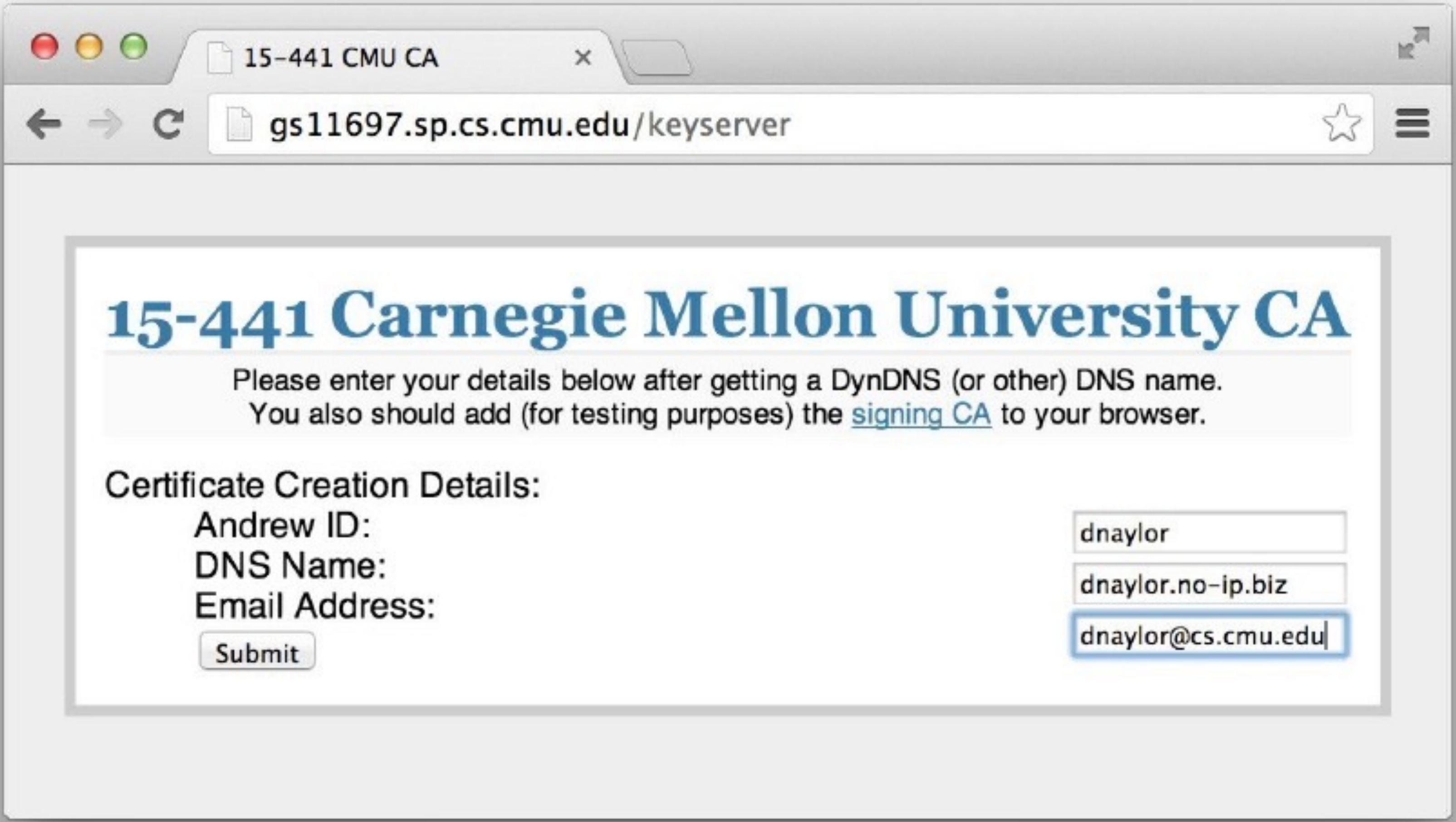
```
[dnaylor@unix3 ~/noip-2.1.9-1]$ ./noip2 -c noip.conf -i 108.17.82.243
IP address detected on command line.
Running in single use mode.
```

Getting a...

## Certificate

# 15-441 Certificate Authority

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



15-441 Carnegie Mellon University CA

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.

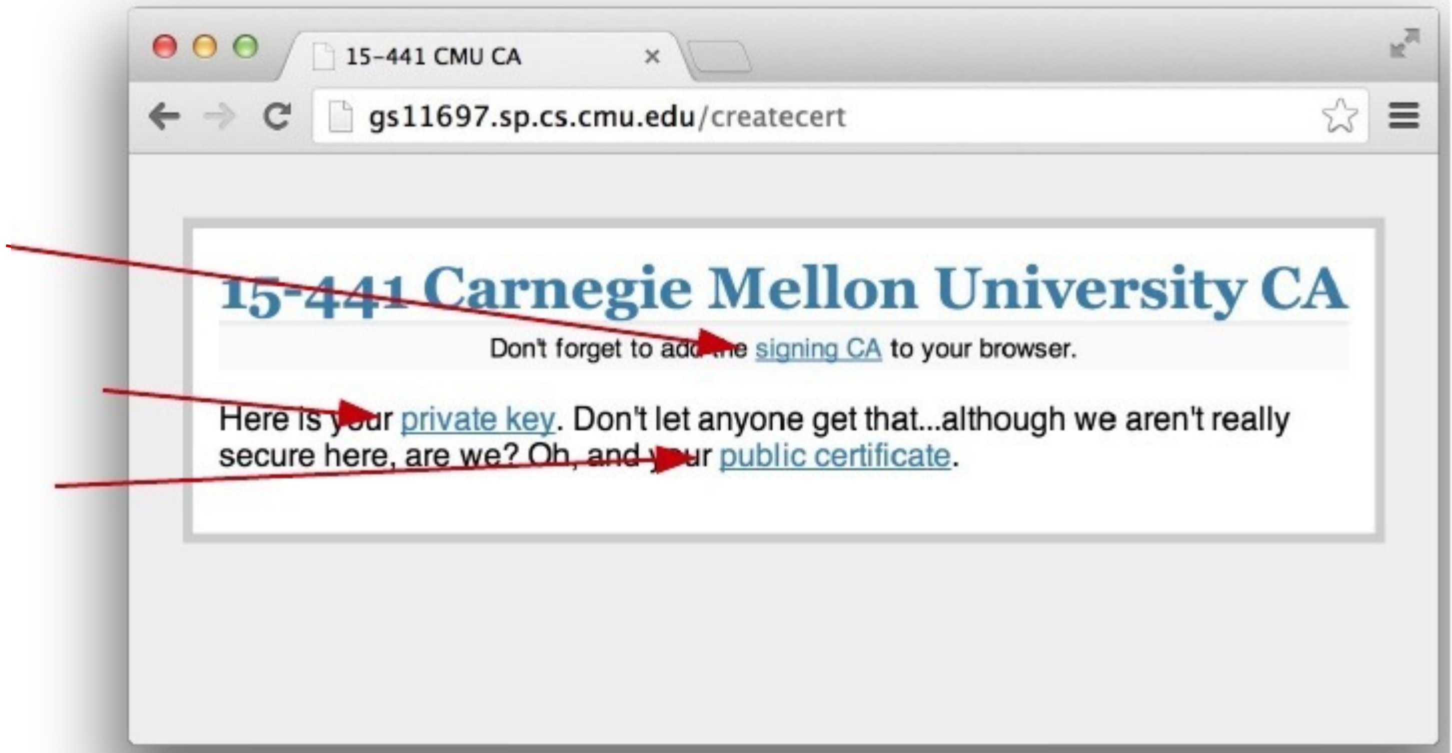
Certificate Creation Details:

| | |
|---|---|
| Andrew ID: | dnaylor |
| DNS Name: | dnaylor.no-ip.biz |
| Email Address: | dnaylor@cs.cmu.edu |

Submit

## You Need 3 Things

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



15-441 CMU CA

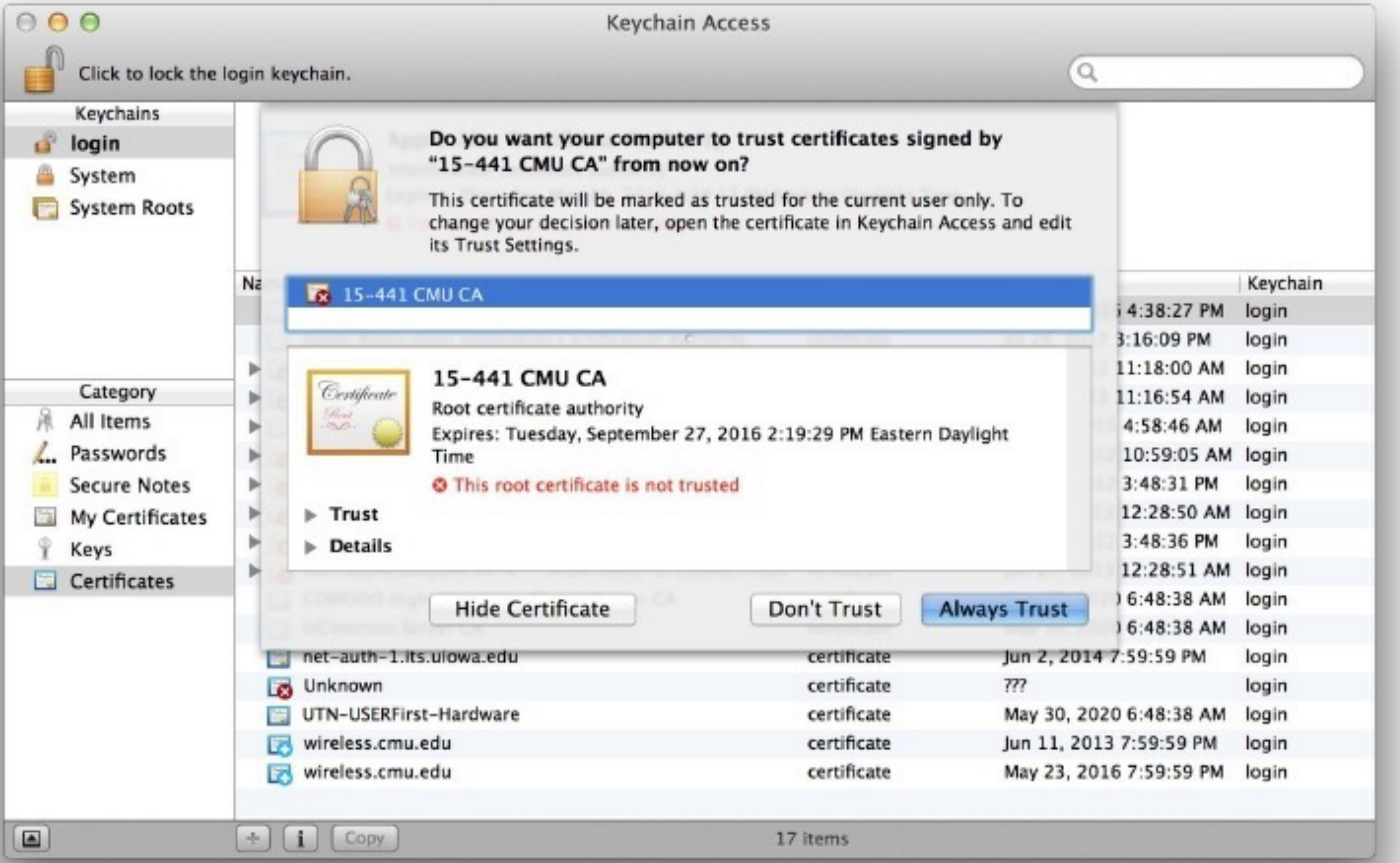
gs11697.sp.cs.cmu.edu/createcert

15-441 Carnegie Mellon University CA

Don't forget to add the signing CA to your browser.

Here is your private key. Don't let anyone get that...although we aren't really secure here, are we? Oh, and your public certificate.

# Add CA Cert to Your System/Browser



**Do you want your computer to trust certificates signed by "15-441 CMU CA" from now on?**

This certificate will be marked as trusted for the current user only. To change your decision later, open the certificate in Keychain Access and edit its Trust Settings.

15-441 CMU CA

**15-441 CMU CA**
Root certificate authority
Expires: Tuesday, September 27, 2016 2:19:29 PM Eastern Daylight Time
This root certificate is not trusted

| | | | |
|---|---|---|---|
| net-auth-1.its.uiowa.edu | certificate | Jun 2, 2014 7:59:59 PM | login |
| Unknown | certificate | ??? | login |
| UTN-USERFirst-Hardware | certificate | May 30, 2020 6:48:38 AM | login |
| wireless.cmu.edu | certificate | Jun 11, 2013 7:59:59 PM | login |
| wireless.cmu.edu | certificate | May 23, 2016 7:59:59 PM | login |

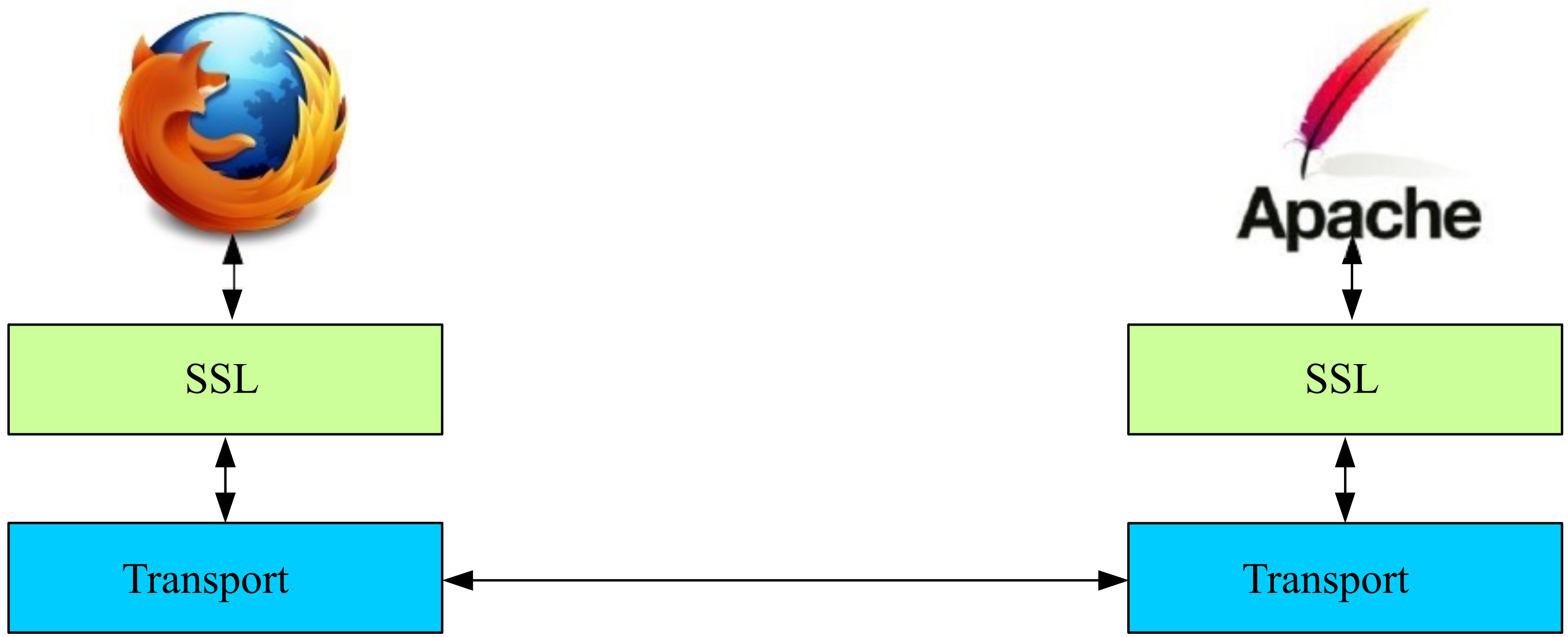
e.g., add to OSX Keychain

## 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.
- 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().)

## SSL

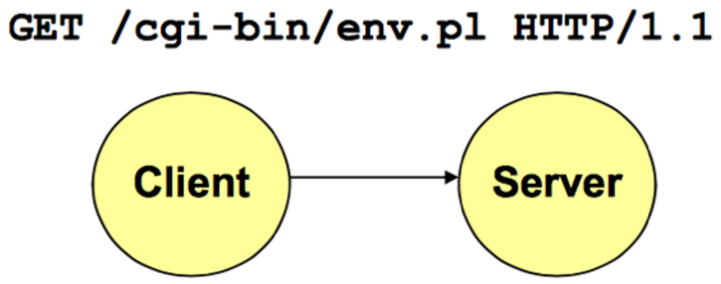
Questions?

## What is CGI?

- A standard method used to generate dynamic content on Web pages and Web applications.
- Provides an interface between the Web server and programs that generate the Web content.
- Usually written in a scripting language.

# Serving Dynamic Content

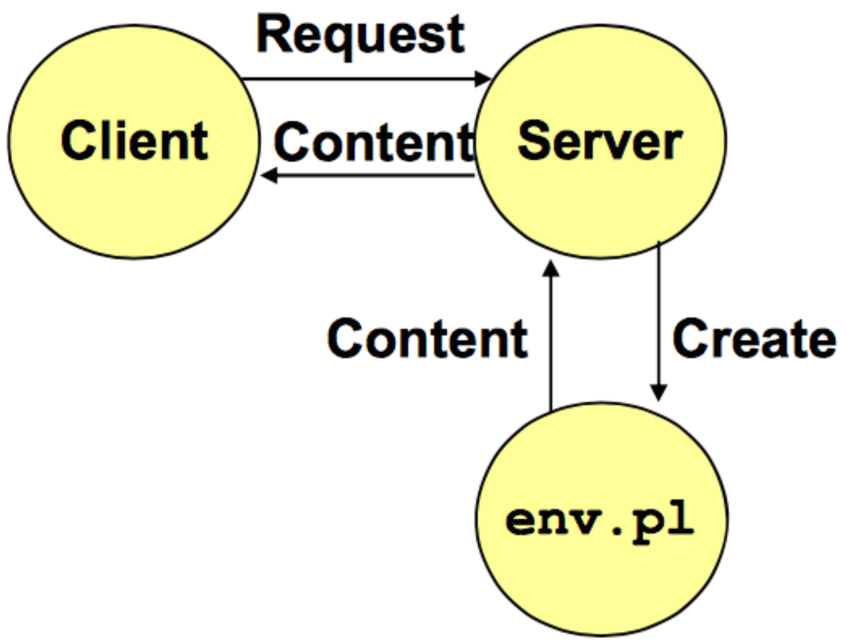
- A Web server that supports CGI can be configured to interpret a URL that it serves as a reference to a CGI script.
- A common convention is to have a `cgi-bin/` directory containing the CGI scripts.



```
GET /cgi-bin/env.pl HTTP/1.1
```

- The server forks a child process and runs the program identified by the URI in that process.

- The server captures the content of the child and forwards it without modification to the client.



## How does the client pass arguments to the server?

- GET: The arguments are appended to the URI can be encoded directly in a URL typed to a browser or a URL in an HTML link.
  - A question mark appended to the URL, followed by param=value pairs.
  - e.g. http://add.com/cgi-bin/adder?1&2

- POST: The arguments are passed in the request body.
  - e.g. name="mark"

## How does the server pass arguments to the cgi program?

- Environment Variables
    - set before execution.
    - passed through exec.

- Request body
    - request body passed to the cgi program's stdin using dup.

# Requirements for LISO

```
REMOTE_ADDR -- taken when accept() call is made
SCRIPT_NAME -- hard-coded/configured application name (virtual path)
SERVER_PORT -- as configured from command line (HTTP or HTTPS port depending)
SERVER_PROTOCOL -- "HTTP/1.1"
SERVER_SOFTWARE -- "Liso/1.0"
GATEWAY_INTERFACE -- “CGI/1.1"

// From request
PATH_INFO
QUERY_STRING
REQUEST_URI
REQUEST_METHOD
CONTENT_LENGTH
CONTENT_TYPE

HTTP_ACCEPT
HTTP_REFERER
HTTP_ACCEPT_ENCODING
HTTP_ACCEPT_LANGUAGE
HTTP_ACCEPT_CHARSET
HTTP_HOST
HTTP_COOKIE
HTTP_USER_AGENT
HTTP_CONNECTION
HTTP_HOST
```

# CGI

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?