Web Proxy

Description:

This program will simulate a web proxy server that will take incoming client HTTP requests, parse the information within the header and make a request to the webserver on behalf of the client for the web objects for the page requested. It will the forward all responses back to client or response with an error message.

./bin/myproxy 9843 bin/forbidden bin/log

Once the servers are running we then must use curl or wget to simulate a web client by requesting a specific web object from a known HTTP or HTTPS webserver.

Curl -x 127.0.0.1:9843/ www.example.com:80/index.html

Wget www.example.com/index.html -e use_proxy=yes -e http_proxy=127.0.0.1:9843

The result of these command would be a copy of the web object requested stored within the current working directory

Port number: 1024 <= port <= 65535

Modules:

- myproxy.c
  - int main(int argc, char **argv)
  - Check if the number of command line arguments provided are accurate and are valid
  - Opens forbidden file for reading and copies data from within into a string array
  - Open log file for writing so all output from server log messages are written to there
  - Initialize locks and Openssl objects, set server sockaddr struct and open socket and bind port to socket
  - Put socket into listening state and wait in infinite loop for incoming client requests
  - Once new client is identified create thread structure for client and call start routine.
  - Each thread will then parse their header make a TCP connection to webserver, formulate a new HTTP request message to webserver, then establish ssl connection with webserver
  - Then it will send HTTP request and wait for the response
  - Once all data has been received the thread will then shutdown ssl connection close TCP connection and free all associated memory for the client.
- Syscall.c/Syscall.h : Wrapper API for error checking system calls
  - int Socket(int family, int type, int protocol)
    - Calls the socket() and checks if it returns an error
  - int Bind(int socket, const struct sockaddr *addr, socklen_t addr_len)
    - Calls the bind() and checks if it returns an error
  - int Connect(int socket, const struct sockaddr *addr, socklen_t addr_len)
    - Calls the connect() and checks if it returns an error
  - int Listen(int socket, int backlog)
    - Calls the listen() and checks if it returns an error
  - int Accept(int socket, struct sockaddr *addr, socklen_t *addr_len)
    - Calls the accept() and checks if it returns an error
  - ssize_t Read(int fd, void *buf, size_t count)
    - Calls the read() and checks if it returns an error
  - ssize_t Write(int fd, const void *buf, size_t count)
    - Calls the write() and checks if it returns an error
  - ssize_t Recvfrom(int sockfd, void *restrict buf, size_t len, int flags, struct sockaddr *restrict src_addr, socklen_t *restrict addrlen)
    - Calls the recvfrom() and checks if it returns an error
  - ssize_t Sendto(int socket, const void *message, size_t length, int flags, const struct sockaddr *dest_addr, socklen_t dest_len)
    - Calls the sendto() and checks if it returns an error

- const char *Inet_ntop(int af, const void *restrict src, char *restrict dst, socklen_t size)
  - Gets the network byte order ip address from struct and converts it to string
- int Inet_pton(int af, const char *restrict src, void *restrict dst)
  - Calls system function inet_ptons and checks if returns an error
- int Setsockopt(int socket, int level, int option_name, const void *option_value, socklen_t option_len)
  - Calls system function setsocketopt() and check if returns an error
- FILE *Fopen(const char *restrict pathname, const char *restrict mode)
  - Calls system function fopen() and check if returns an error
- int Fclose(FILE *stream)
  - Calls system function fclose() and check if returns an error
- size_t Fread(void *restrict ptr, size_t size, size_t nmemb, FILE *restrict stream)
  - Calls system function fread() and check if returns an error
- size_t Fwrite(const void *restrict ptr, size_t size, size_t nitems, FILE *restrict stream)
  - Calls system function fwrite() and check if returns an error
- void Mkdir(char *s)
  - Calls mkdir() and checks if it returns an error. If not it will recursively make a directory
- int Getsockname(int sockfd, struct sockaddr *restrict addr, socklen_t *restrict addrlen)
  - Calls getsockname() to get socket information of the passed in socket and stores this info into sockaddr struct. Error checks this system call as well.
- int Getaddrinfo(const char *node, const char *service, const struct addrinfo *hints, struct addrinfo **res)
  - Does a DNS lookup for an IP address within the sockaddr struct specified and returns a list of information for the caller to use
- int Getnameinfo(const struct sockaddr *restrict addr, socklen_t addrlen, char *restrict host, socklen_t hostlen, char *restrict serv, socklen_t servlen, int flags)
  - Does a reverse DNS lookup for a given IP address.
- SSL_CTX *SSL_CTX_NEW(const SSL_METHOD *method)
  - Creates a SSL context object
- int SSL_CTX_SET_CIPHER_LIST(SSL_CTX *ctx, const char *str)
  - Sets a list of Cipher for the SSL context
- SSL *SSL_NEW(SSL_CTX *ctx)
  - Creates a new SSL object
- int SSL_SET_FD(SSL *ssl, int fd)
  - Sets the ssl object to a given socket
- int SSL_SET_TLSEXT_HOST_NAME(SSL *s, const char *name)
  - Sets the SNI for a ssl conneciton
- int SSL_WRITE(SSL *ssl, const void *buf, int num)
  - Write through a SSL connection to the end peer
- int SSL_READ(SSL *ssl, void *buf, int num)
  - Reads incoming data through a SSL connection from an end peer

- misc.c/misc.h (used from previous Lab)
  - char* getpath(char* s)
    - Parses through a string to extract the path and returns it to the calling function.
  - int checkipv4(char* s)
    - check if the string which is an ipv4 address have 4 octets and has integer values between 0-255
  - int checknum(char* s)
    - Checks if the string passed is a valid integer number, if not it returns -1, else it returns the value.
  - void print_log (FILE *stream, char *client_ip, char *request_line, char *code, int bytes)
    - Prints a log output from the proxy server for client requests
  - int gethost(char **host, char *header);
    - Parses header and gets hostname

    - void getagent(char *agent, char *header);
        - Parses header and gets user agent
    - void getpath(char agent, char **path, char *header);
        - Parses header and gets path
    - void getrequest(char **request, char *header);
        - Parses header and gets request type
    - void getport(int *port, char *header, int host_len);
        - Parses header and gets port number
    - void getcode(char **code, char *header);
        - Parses response and gets return code
    - void get_line_request(char **line, char *header);
        - Parses header and gets first line of request
    - char **get_forbidden_sites(char *filename);
        - Opens forbidden sites file and populates string array containing this information.
- Proxy_manager.c/ Proxy_manager.h
    - int Resolve_Connect(char *hostname, int port)
        - Either does a DNS look up for a hostname and uses returns sockaddr struct to make a TCP connection or uses given ip address to make connection with webserver
    - void Send_Error(int clientfd, char *error)
        - Sends specific HTTP error response messages back to the client
    - int website_check(struct addrinfo *res, char *hostname)
        - Check the list of forbidden sites for a clients request. If found within the string array and 403 error is returned
    - void sigintHandler(int sig_num)
        - Handles the incoming CTRL+C signal for updating the list of forbidden sites
    - void free_Thread_Memory(struct thread_info **thread_list)
        - Frees all memory for a specific thread.

Test Case:

```
./bin/myproxy 9843 bin/forbidden bin/log
```

```
curl -x http://127.0.0.1:9843/ www.troyhunt.com:80/heres-why-your-static-website-needs-https/
```

```
./bin/myproxy 9843 bin/forbidden bin/log
```

```
wget www.troyhunt.com:80/heres-why-your-static-website-needs-https/ -e use_proxy=yes -e http_proxy=127.0.0.1:9843
```

```
./bin/myproxy 9843 bin/forbidden bin/log
```

```
curl -x http://127.0.0.1:9843/ news.ycombinator.com:80/item?id=22146291
```

(100MB binary file)

```
./bin/myproxy 9843 bin/forbidden bin/log
```

```
curl -x http://127.0.0.1:9843/ speed.hetzner.de:443/100MB.bin
```

(No port specified)

```
./bin/myproxy 9843 bin/forbidden bin/log
```

```
wget en.wikipedia.org/wiki/Main_Page -e use_proxy=yes -e http_proxy=127.0.0.1:9843
```

(Multithreading)

Also tested server for 500 simultaneous client request 50 times to the same webserver

Shortcomings

None that have been identified.

Citations

Used code from https://curl.se/libcurl/c/opensslthreadlock.html to initialize a set number of locks for the openssl library to use and set specific call back function for the library to use for maintaining concurrency in this multithreaded environment. Snipper of code is at the top of the myproxy.c file.