Refactoring a Project

Let's work on some messy code that needs restyling

What we will do

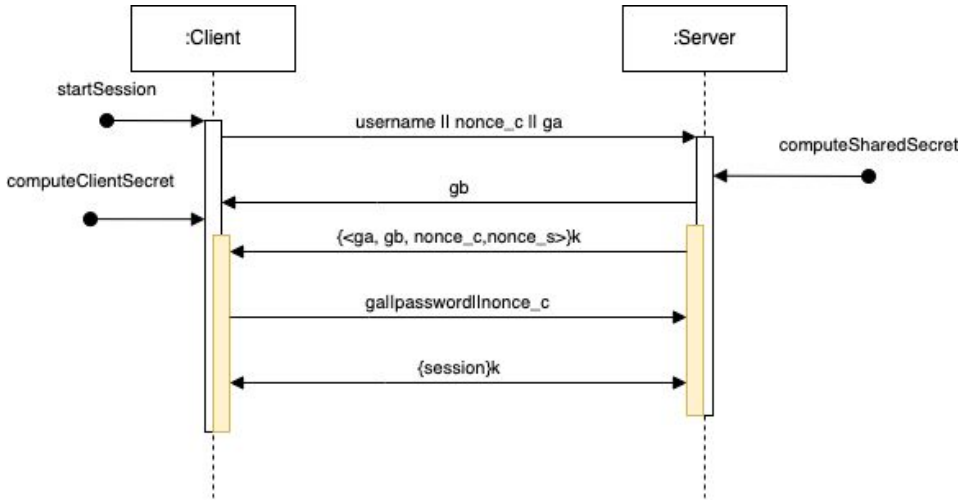
The project

The project entails a secure Client-Server application utilizing a shared secret computed through the Diffie-Hellman algorithm.

The client initiates communication by sending its public key, nonce, and username for login.

In response, the server sends its own public key.

Both the client and server then establish a shared session for encrypted communication



The messy code

Let's break down the function into its individual components.

- **Parse client session:** username || nonce_c || ga
- **Diffie-Hellman:** Compute parameters g_b and send it to client
- **Compute shared secret:** using g_a client and g_b server
- **Encrypt server session**: <g_a, g_b, nonce_c, nonce_s>
- **Send server session**: to client
- **Receive client session**: <g_a, password, nonce_c>
- **Decrypt client session**
- **Parse client session**: to assign session to the user
- **Login:** find the user and assign session

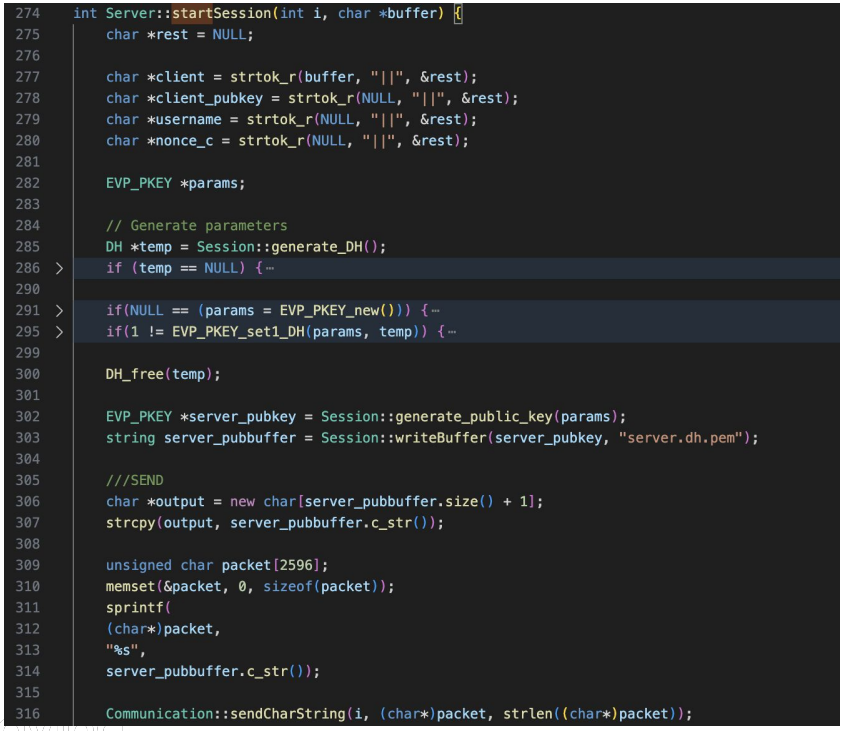
Over 300 lines of code in a single function!!!

davinci
Mirko Di Lucia

Code Smells

Here's some of the smells we found

- Long Method
- Large Class
- Divergent Change (God Class)
- Data Clumps
- Shotgun Surgery
- Feature envy
- Comments
- Duplicated code



```
274  int Server::startSession(int i, char *buffer) {
275      char *rest = NULL;
276
277      char *client = strtok_r(buffer, "||", &rest);
278      char *client_pubkey = strtok_r(NULL, "||", &rest);
279      char *username = strtok_r(NULL, "||", &rest);
280      char *nonce_c = strtok_r(NULL, "||", &rest);
281
282      EVP_PKEY *params;
283
284      // Generate parameters
285      DH *temp = Session::generate_DH();
286      if (temp == NULL) {…
290
291      if(NULL == (params = EVP_PKEY_new())) {…
295      if(1 != EVP_PKEY_set1_DH(params, temp)) {…
299
300      DH_free(temp);
301
302      EVP_PKEY *server_pubkey = Session::generate_public_key(params);
303      string server_pubbuffer = Session::writeBuffer(server_pubkey, "server.dh.pem");
304
305      ///SEND
306      char *output = new char[server_pubbuffer.size() + 1];
307      strcpy(output, server_pubbuffer.c_str());
308
309      unsigned char packet[2596];
310      memset(&packet, 0, sizeof(packet));
311      sprintf(
312      (char*)packet,
313      "%s",
314      server_pubbuffer.c_str());
315
316      Communication::sendCharString(i, (char*)packet, strlen((char*)packet));
```

Let’s refactors

With this simple method

- **Expand:**
  - Created new structs for Session
    - BaseSession
    - ServerSession
    - ClientSession
  - Created new class:
    - SecureSession
    - SecureServerSession
    - SecureClientSession
  - Write test for the new functions
  - Move duplicated code to functions

```
10  struct BaseSession {
11      char *ga;
12      char *gb;
13
14      unsigned char *secret;
15
16      char *nonce_c;
17      char *nonce_s;
18  };
19
20  struct DatabaseSession {
21      unsigned char *key;
22      unsigned char *iv;
23  };
24
25  struct UserSession : BaseSession {
26      char *client;
27      EVP_PKEY *client_pubkey;
28      char *username;
29      char *password;
30  };
31
32  struct ClientSession : BaseSession
33      char *client;
34      char *client_pubkey;
35      char *username;
36      char *password;
37
38      ClientSession(char *username, char *password) {
39          this->username = username;
40          this->password = password;
41      }
42  };
```

```
38  void ServerSecureSession::parseSessionUsername(char *buffer, UserSession *&session) {
39      char *rest;
40
41      char *client = strtok_r( str: buffer, sep: "||", lasts: &rest);
42      char *username = strtok_r( str: nullptr, sep: "||", lasts: &rest);
43      char *nonce_c = strtok_r( str: nullptr, sep: "||", lasts: &rest);
44      char *clientPubKey = strtok_r( str: nullptr, sep: "||", lasts: &rest);
45
46      session->client = (char *) malloc( size: strlen( s: client) + 1);
47      memccpy( dst: session->client, src: client, c: 0, n: strlen( s: client));
48
49      session->client_pubkey = loadPublicKeyFromCharArray(clientPubKey);
50
51      session->username = (char *) malloc( size: strlen( s: username) + 1);
52      memccpy( dst: session->username, src: username, c: 0, n: strlen( s: username));
53
54      session->nonce_c = (char*) malloc( size: strlen( s: nonce_c) + 1);
55      memccpy( dst: session->nonce_c, src: nonce_c, c: 0, n: strlen( s: nonce_c));
56  }
```

davinci
Mirko Di Lucia

Let’s refactors

With this simple method

- **Migrate:**
  - Rewrite the large method by instantiate new classes
  - Call functions created in the previous step

```
ServerSecureSession* ServerSecureSession::startSession(int socket, char *buffer) {
    auto *session = new UserSession();
    parseSessionUsername(buffer, &session);

    EVP_PKEY *params = initializeDH();
    if (params == nullptr) {
        Logger::log( level: ERROR, message: "Error initializing DH parameters.\n");
        exit(1);
    }

    char *serverPubKeyFileContent;
    EVP_PKEY *server_pubkey = generatePublicKey(params);
    savePublicKeyAndReadIt( fileName: "server.dh.pem", pubkey: server_pubkey, &serverPubKeyFileContent);

    // Send public key to client g^b
    unsigned char outPacket[2596];
    memset( b: &outPacket, c: 0, len: 2596);
    snprintf( str: (char*)outPacket, size: 2596, format: "%s", serverPubKeyFileContent);
    Communication::sendString(socket, outPacket, strlen( s: (char*)outPacket));

    char *clientPubKeyFileContent;
    savePublicKeyAndReadIt( fileName: "client.dh.pem", pubkey: session->client_pubkey, &clientPubKeyFileContent);
    unsigned char* sharedSecret;
    if (computeSharedSecret(server_pubkey, session->client_pubkey, &sharedSecret) < 0) {
        Logger::log( level: ERROR, message: "Error computing shared secret.\n");
        exit(1);
    }
```

Let’s refactors

With this simple method

- **Contract:**
  - Remove old code in the large methods

```
    session->secret = (unsigned char *)malloc( size: SHA256_DIGEST_LENGTH);
    memccpy( dst: session->secret, src: sharedSecret, c: 0, n: SHA256_DIGEST_LENGTH);

    // Send {<g^a||g^b||nonce_c||nonce_s}
    Nonce::newNonce( &: session->nonce_s);

    memset( b: &outPacket, c: 0, len: 2596);
    snprintf(
            str: (char*)outPacket,
            size: 2596,
            format: "%s||%s||%s||%s",
            serverPubKeyFileContent,
            clientPubKeyFileContent,
            session->nonce_s,
            session->nonce_c
    );

    Communication::sendEncryptedString(socket, sharedSecret, (char*)outPacket, strlen( s: (char*)outPacket), session->
    char* buffer_received = Communication::receiveEncryptedString(socket, sharedSecret);

    parseSessionPassword( buffer: buffer_received, &: session);

    if (1 == ServerSecureSession::checkSessionIntegrity( &: session, clientPubKey: clientPubKeyFileContent, serverPubKey:
        return new ServerSecureSession(session);
    }

    return nullptr;
}
```

Thanks

Refactoring helps in rewriting the code to enhance reusability between the client and server by introducing a class hierarchy and the SecureSession class concept.

Test-Driven Development (TDD) aids in quickly detecting bugs, allocation errors, and broken strings, and ensures nonce integrity checks.

This particular refactor was notably complex because of the method's shared nature between the client and server

Mirko Di Lucia
Email: dlmrk95@gmail.com
https://www.linkedin.com/in/mirko-di-lucia-3764701b1/