

Forum:[Hack Forums](#)

Thread: [\[C++ | OpenSSL\] Logging into Steam Website](#)

Bobface - October 24, 2015, 3:21 am

Hello everyone,

I'm trying to log into the Steam website (steampowered.com) using C++.

The login process is broken down into two steps:

/login/getrsakey/: The server sends you an RSA public key with which you have to encrypt your password and then in a second request send your encrypted password.

/login/dologin/: Send your username with your encrypted password and the login is finished.

To encrypt my password I'm using OpenSSL. This is the code I have so far:

```
Code string CCrypto::encryptRSAFromModExp(string SMod, string SExp, string SMsg)
{
    const char* Exp = SExp.c_str();
    const char* Mod = SMod.c_str();
    const char* Msg = SMsg.c_str();

    RSA *pubkey = RSA_new();

    int len = BN_hex2bn(&pubkey->n, Mod);
    if (len == 0 || Mod[len])
        cout << "Invalid Modulus" << endl;
    BN_hex2bn(&pubkey->e, Exp);

    unsigned char* out = new unsigned char[RSA_size(pubkey)];

    if (RSA_public_encrypt(strlen(Msg), (const unsigned char *) Msg, out, pubkey,
RSA_PKCS1_OAEP_PADDING) <= 0)
    {
        cout << "Error at RSA_public_encrypt" << endl;
        return "";
    }

    string ret(reinterpret_cast<char*>(out), RSA_size(pubkey));

    ret = CCrypto::Base64Encode(ret);

    return ret;
}
```

However, when I send the password back it tells me "Invalid Login-Data".

This is the original JavaScript-Code that gets executed when you normally log into Steam:

```
Code var RSA = {
```

```
  getPublicKey: function( $modulus_hex, $exponent_hex ) {
    return new RSAPublicKey( $modulus_hex, $exponent_hex );
  },

  encrypt: function($data, $pubkey) {
    if (!$pubkey) return false;
    $data = this.pkcs1pad2($data,($pubkey.modulus.bitLength()+7)>>3);
    if(!$data) return false;
    $data = $data.modPowInt($pubkey.encryptionExponent, $pubkey.modulus);
    if(!$data) return false;
    $data = $data.toString(16);
    if(($data.length & 1) == 1)
      $data = "0" + $data;
    return Base64.encode(Hex.decode($data));
  },

  pkcs1pad2: function($data, $keysize) {
    if($keysize < $data.length + 11)
      return null;
    var $buffer = [];
    var $i = $data.length - 1;
    while($i >= 0 && $keysize > 0)
      $buffer[--$keysize] = $data.charCodeAt($i--);
    $buffer[--$keysize] = 0;
    while($keysize > 2)
      $buffer[--$keysize] = Math.floor(Math.random()*254) + 1;
    $buffer[--$keysize] = 2;
    $buffer[--$keysize] = 0;
    return new BigInteger($buffer);
  }
};
```

```
var encryptedPassword = RSA.encrypt( password, pubKey );
```

Can anyone spot my mistake or maybe someone has done this already?

Thanks for your help!

topLevel - October 24, 2015, 8:57 pm

Make sure you are handling endian byte ordering. windows uses little endian where openssl functions use big endian.

You must reverse the byte ordering of the integers in the key. To handle messages you must reverse each block separately.