PB173 - Tématický vývoj aplikací v C/C++ (Jaro 2016)

Skupina: Aplikovaná kryptografie a bezpečné programování

https://is.muni.cz/auth/el/1433/podzim2013/PB173/index.qwarp?fakulta=1433;obdobi=5983;predmet=734514;prejit=2957738;

Petr Švenda svenda@fi.muni.cz

Konzultace: A.406, Pondělí 15-15:50

CROCS

Centre for Research on Cryptography and Security

NETWORK COMMUNICATION

Sockets

- Plain sockets (Windows Sock library…)
- UDP/TCP/TLS in QT (QT += network)
  - QTcpSocket, QUdpSocket, QSslSocket
  - M. Jaros
- (QT in Visual Studio)
  - http://portfolio.delinkx.com/files/Qt.pdf
  - VS Add in: http://www.bogotobogo.com/Qt/Qt5_Visual_Studio_Add_in.php
- (QT in NetBeans)
  - https://netbeans.org/kb/72/cnd/qt-applications.html

Networking via QT

- Presentation given by M. Jaros
- Slides:
  - https://github.com/mijaros/QtNetworks/raw/master/presentation/presentation.pdf
- Repository with examples
  - https://github.com/mijaros/QtNetworks

ANOTHER OPTION

Network communication with sockets

- Many different libraries
  - e.g., René Nyffenegger class wrappers
  - https://github.com/ReneNyffenegger/development_misc/tree/master/c++/socket
  - build atop of Windows Sock library <WinSock2.h>
- Client & Server mode of communication
  1. Server run and is waiting for client(s)
  2. Client connects to server (IP, port)
  3. Server creates new thread for every new client
  4. Client sends data to server (SendLine("data"))
  5. Server receives data and respond to client

Socket client and server

```
class SocketClient : public Socket {
public:
  SocketClient(const std::string& host, int port);
};

class SocketServer : public Socket {
public:
  SocketServer(int port, int connections, TypeSocket type=BlockingSocket);
  Socket* Accept();
};
```

```
enum TypeSocket {BlockingSocket, NonBlockingSocket};

class Socket {
public:

  virtual ~Socket();
  Socket(const Socket&);
  Socket& operator=(Socket&);

  int recvtimeout(SOCKET s, char *buf, int len, int timeout);

  std::string ReceiveLine(int timeout = 0);
  std::string ReceiveBytes();

  std::string ReceiveResponse(std::string endSeq, int timeout);
  int ReceiveLineToFile(std::string filePath, int timeout, int* pWrittenValues = NULL);

  void   Close();

  // The parameter of SendLine is not a const reference
  // because SendLine modifes the std::string passed.
  void   SendLine (std::string);

  // The parameter of SendBytes is a const reference
  // because SendBytes does not modify the std::string passed
  // (in contrast to SendLine).
  void   SendBytes(const std::string&);
};
```

Start new proxy server

```
void startProxy(int proxyPort) {
   // TERMINATE PREVIOUS THREAD
   if (m_pProxyThread != NULL) {
      m_bStopThread = TRUE;
      m_pSocketServer->Close();
      Sleep(1000);
      delete m_pSocketServer;
   }

   m_pSocketServer = new SocketServer(atoi(proxyPort), 1);

   m_bStopThread = FALSE;
   m_pProxyThread = AfxBeginThread(StartProxy, (void*) m_pSocketServer,
                                   THREAD_PRIORITY_NORMAL);
}
```

Waiting for new client

```
UINT StartProxy(void* a) {
    SocketServer* pSocketServer = (SocketServer*) a;
    cout << "Proxy started";

    while (1) {
        try {
            Socket* s=pSocketServer->Accept();
            cout << "\nNew connection detected and waiting for service";
            // TERMINATE IF REQUIRED
            if (m_bStopThread) break;
            m_pProxyThread = AfxBeginThread(Answer, (void*) s,
                                            THREAD_PRIORITY_NORMAL);
        }
        catch(...) {
            // TERMINATE IF REQUIRED (POSSIBLE REASON FOR EXCEPTION)
            if (m_bStopThread) break;
            else cout << "Exception in pSocketServer->Accept";
        }
    }
    return 0;
}
```

Connect client, send message

```
UINT clientCommunicate(void* a) {
    SocketClient*    pCardSocket = NULL;
    // Connect to socket
    try {
        pCardSocket = new SocketClient("127.0.0.1", 4001);
    }
    catch (...) {
        cout << "Fail to connect to socket";
        pCardSocket = NULL;
    }
    if (pCardSocket != NULL) {
        // Send command to server
        pCardSocket->SendLine(cmd);
        // Wait for response
        string = pCardSocket->ReceiveLine();
        // TODO: Parse response
    }
    if (pCardSocket) delete pCardSocket;
    return 0;
}
```

Obtain message from socket, respond

```
UINT Answer(void* a) {
   Socket* pSocket = (Socket*) a;
   while (1) {
      Sleep(100);
      if (pSocket != NULL) {
         std::string r = pSocket->ReceiveLine();
         if (value != "") {
            // TODO: Parse socket input

            // TODO: Create response
             std::string  cmd("some response");
             pSocket->SendLine(cmd);
         }
      }
      else return -1;
   }
    // Delete served socket
   delete pSocket;
   return 0;
}
```

Assignment – basic networking

- Implement network communication between client and server
  - Login, list of users…
  - Use only simple TCP communication (no TLS yet)
- Setup test environment for integration test with network
  - Test on localhost
  - Test between multiple computers (same network)
  - Perform tests

Submissions, deadlines

- Upload application source codes as single zip file into IS Homework vault (Crypto - 7. homework (Network))
- DEADLINE 18.4. 12:00
  - 0-10 points assigned