
Subject: Re: SSL server crash

Posted by [mirek](#) on Sat, 10 Sep 2011 08:16:24 GMT

[View Forum Message](#) <> [Reply to Message](#)

This might work as long as you have single working thread. Otherwise

```
if (!clients[ci].sock.IsOpen() || clients[ci].sock.IsError() || clients[ci].sock.IsEof()
```

you can get "Remove" at '|'|.

It is not a very good code in any case.

Array does not change many things as compered to Vector.

The correct code would be something line:

```
class TestServer{
volatile Atomic stop;
Vector<client_data> clients;
    Mutex mtx;

int WaitForInput(int _timeout_ms);
void Worker();
public:
typedef TestServer CLASSNAME;
void Start();
void Stop() {AtomicWrite(stop, true);}
TestServer() {AtomicWrite(stop, false);}
};

int TestServer::WaitForInput(int timeout_ms)
{
    fd_set set;
    struct timeval tval;
    int max = 0;

    if (clients.GetCount() <= 0) return -1;

    tval.tv_sec = timeout_ms / 1000;
    tval.tv_usec = 1000 * (timeout_ms % 1000);

    mtx.Enter();
    FD_ZERO(&set);
    for (int i = 0; i < clients.GetCount(); i++){
        int tmp = clients[i].sock.GetSocket();
        if (tmp > max) max = tmp;
```

```

    FD_SET(tmp, &set);
}

    mtx.Leave();

if (select(max+1, &set, NULL, NULL, &tval) > 0){
    mtx.Enter();
    for (int i = 0; i < clients.GetCount(); i++){
        if (FD_ISSET(clients[i].sock.GetSocket(), &set)) return i;
    }
}

    else
        mtx.Enter();

return -1;
}

void TestServer::Worker()
{
    while(!Thread::IsShutdownThreads() && !AtomicRead(stop))
    {
        int ci = WaitForInput(1000);
        if (ci >= 0){
            if (!clients[ci].sock.IsOpen() || clients[ci].sock.IsError() || clients[ci].sock.IsEof()){
                Cout() << "Client no " << ci << " (" << FormatIP(clients[ci].ip) << ") closed connection\n";
                clients[ci].sock.Close();
                clients.Remove(ci);
            }else{
                Cout() << "[" << FormatIP(clients[ci].ip) << "]" " << clients[ci].sock.Read() << "\n" ;
            }
        }else{
            for (int i = 0; i < clients.GetCount(); i++){
                clients[i].sock.Write(Format("%`: message to client no %d", GetSysTime(), i));
            }
        }

        mtx.Leave();
    }
}

void TestServer::Start()
{
    Socket server;
    #if 1
    SSLContext context;

    if (!context.Create(SSLv3_server_method)){

```

```

    Cout() << "Can not create context\n";
    return;
}

if (!context.UseCertificate(LoadFile(ConfigFile("servercert.pem")),
LoadFile(ConfigFile("serverkey.pem")), false)){
    Cout() << "Certificate and key are different!\n";
    return;
}

if(!SSLServerSocket(server, context, 11111, true, 5, true)){
#else
    if(!ServerSocket(server, 11111)){
#endif
        Cout() << "Can not start server\n";
        return;
    }

    Thread().Run(THISBACK(Worker));

    Cout() << "Waiting for connections\n";

    while(!Thread::IsShutdownThreads() && !AtomicRead(stop)){
        client_data new_client;
        if (server.Accept(new_client.sock, &new_client.ip)){
            Cout() << "Client no " << clients.GetCount() << " from " << FormatIP(new_client.ip) << "\n";
            mtx.Enter();
            clients.Add(new_client);
            mtx.Leave();
        }else{
            Cout() << "+\n";
        }
    }
}

TestServer server;

void Stop(int sig)
{
    Cout() << "\nstopping, please wait...\n";
    server.Stop();
}

CONSOLE_APP_MAIN
{
    signal(SIGINT, Stop);

```

```
server.Start();  
}
```

I do not quite like the structure, as we now leave WaitForInput with mutex locked, but I believe it is now correct.

Mirek
