

Port Scanning

by Randy Charles Morin

I just started working at a new startup. This time, the start is from scratch. I actually wrote the first few hundred lines of code myself, the original Software Development Plan and System Architecture Document. It's quite fun. My last dot-com home went from 50 cents to \$345 and is now sitting at \$9 and change. A real roller coaster! I can only wish the same. I'll keep you all posted.

Anyhow, the downside of being in very early is that I've acquired the privilege of configuration our Internet Security and Acceleration (ISA) server. We had a couple of scares where we thought somebody might be trying to figure out what we are doing. From my own history with a dot-com roller coaster and from the history of my workmates, we put the two and two together and figured that between the hackers hacking unsuspecting sites and the pirates that exist at previous employs, the need for secure Internet access was very important. At first, I thought I was paranoid and worrying needlessly about outside attacks on our network. Then we got a virus thru open IIS ports. Yikes!

We setup the ISA server and shutdown all external connections to our internal network. But how do you test Internet connectivity? How do you prove that your firewall is really secure? This is the basis of the article. I wrote a simple little command-line utility that does a port scan on your server to determine open ports.

A common technique used for determining the vulnerability of a network or computer is to do a port scan. The port scan involves attempts to connect over a large range of well known ports and then trying to determine from the results of the scan, what the vulnerability of the computer are. The result a secure computer is looking for is the inability of external hackers to connect to any ports. This resembles the result of a computer that is not at all on the Internet.

I decided to write a command-line utility that took one parameter, a computer name or IP address and attempt a large port scan on the computer. My first attempt was using only my socket template class and using a simple loop, I attempted to connect to one port at a time.

Listing 1: class Socket

```
////////////////////////////////////  
//  
// socket.h: implementation of the Socket class.  
// Copyright 2000 by Randy Charles Morin  
// You have unlimited ability to distribute and modify this source,  
// but this legal notice must remain intact and the Socket class must  
// remain within the kbcafe namespace.  
//  
////////////////////////////////////  
  
#ifndef KBCAFE_SOCKET_H  
#define KBCAFE_SOCKET_H  
  
#include <iostream>  
#include <exception>  
#include <string>  
#include "winsock.h"
```

```
namespace
{
class WSAInit
{
public:
    WSAInit()
    {
        WORD w = MAKEWORD(1,1);
        WSADATA wsadata;
        ::WSAStartup(w, &wsadata);
    };

    ~WSAInit()
    {
        ::WSACleanup();
    };

} instance;
}

namespace kbcafe
{
class SocketException : public std::exception
{
    std::string m_str;
public:
    SocketException()
        :m_str("Socket exception")
    {}

    SocketException(const std::string & str)
    {
        m_str = std::string("Socket exception:")+str;
    }
    virtual const char * what() const throw()
    {
        return m_str.c_str();
    }
};

class Socket
{
public:
    SOCKET m_socket;
    Socket();
    virtual ~Socket();
    void Connect(const std::string & strAddress, const std::string & protocol);
    void Connect(const std::string & strAddress, int port);
    virtual std::string Response();
    virtual void Write(const std::string & str);
};

inline Socket::Socket()
    :m_socket(NULL)
{
}

inline Socket::~~Socket()
{
    ::closesocket(m_socket);
};

inline void Socket::Connect(const std::string & strAddress, int port)
{
    hostent * host;
    in_addr inaddr;
    inaddr.s_addr = ::inet_addr(strAddress.c_str());
    if (inaddr.s_addr == INADDR_NONE)
```

```
{
    host = ::gethostbyname(strAddress.c_str());
}
else
{
    host = ::gethostbyaddr((const char *)&inaddr,
        sizeof(inaddr), AF_INET);
}

if (host == NULL)
{
    throw SocketException("invalid SMTP server");
}

m_socket = ::socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (m_socket == INVALID_SOCKET)
{
    throw SocketException("socket invalid");
}

sockaddr_in sa;
sa.sin_family = AF_INET;
sa.sin_port = ::htons(port);
sa.sin_addr.s_addr = *((u_long*)host->h_addr_list[0]);
if (::connect(m_socket, (sockaddr *)&sa, sizeof(sa)) < 0)
{
    throw SocketException("connection to host failed");
};
};

#endif // !defined(AFX_SOCKET_H__EEA38B20_79F9_11D4_9C2B_444553540002__INCLUDED_)
```

A subset of my socket class is presented here. Only the code that we need in this example is shown. In the main body of my utility, I looped thru the possible ports and reported the result of calling the Connect method with each port. The web address was specified on the command-line as the first input parameter.

I could then translate the web name to an IP address using gethostbyname and gethostbyaddr.

```
hostent * host;
in_addr inaddr;
inaddr.s_addr = ::inet_addr(strAddress.c_str());
if (inaddr.s_addr == INADDR_NONE)
{
    host = ::gethostbyname(strAddress.c_str());
}
else
{
    host = ::gethostbyaddr((const char *)&inaddr,
        sizeof(inaddr), AF_INET);
}
```

Then I attempt to connect using the socket and connect socket library functions.

```
m_socket = ::socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (m_socket == INVALID_SOCKET)
{
    throw SocketException("socket invalid");
}

sockaddr_in sa;
sa.sin_family = AF_INET;
sa.sin_port = ::htons(port);
sa.sin_addr.s_addr = *((u_long*)host->h_addr_list[0]);
if (::connect(m_socket, (sockaddr *)&sa, sizeof(sa)) < 0)
```

```

    {
        throw SocketException("connection to host failed");
    };

```

The socket function creates a new end-point for use in a socket connection. The connect function attempt to bind the socket with an end-point at the specified remote IP address.

This attempt was fruitless as the port scan was taking several seconds to determine if a port was connectable or not. Since I wanted to connect to several hundred, if not thousand ports, the performance would make my utility next to useless.

I realized that for this utility to be useful, I would require multiple threads port pinging the server simultaneously. With say 100 or so threads constantly trying to connect, I could reduce the run time of the scan from hours to minutes.

Listing 2: Portscan

```

// portscan.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

namespace
{
    std::string name;
};

class myconsumer : public kbcafe::consumer<int>
{
public:
    myconsumer(int n)
        :kbcafe::consumer<int>(n)
    {
    }

    void consume(int & i)
    {
        if (i%100 == 0)
        {
            std::cout << "Port = " << i << std::endl;
        }

        try
        {
            kbcafe::Socket socket;
            socket.Connect(name, i);
            std::cout << "Connected: " << name << " Port "
                << i << std::endl;
        }
        catch(kbcafe::SocketException &)
        {
        };
    }
};

int main(int argc, char ** argv)
{
    if (argc != 2)
    {
        std::cout << "Usage:\t portscan ip" << std::endl;
        return 0;
    }
    name = argv[1];

    myconsumer c(100);
    c.start();
    for (int j=0;j<2048;j++)
    {

```

```
        c.produce(j);
    }
    c.sleep(120000);
return 0;
}
```

I previously had written a consumer class that would allow this type of multithreaded behavior. I'll discuss how the consumer class works in a future article, as it is more complex that deserves an entire article on its own. But the way the consumer class works, is you specify the template member to be the type of object that the consumer and producer will be sharing in the model. In this case I want to pass port numbers to the consumers and decided to represent the port number by using the integer as the template type parameter.

```
kbcafe::consumer<int>
```

The next step is to implement a derived version of this class (see myconsumer class) based on this template and type parameter.

```
class myconsumer : public kbcafe::consumer<int>
```

You create a new pool of consumer threads by constructing on object of this type and calling the start method.

```
myconsumer c(100);
c.start();
```

You can add things to the consumer queue, by calling the consumer.produce method. In this case, I want to add 2048 port numbers.

```
for (int j=0;j<2048;j++)
{
    c.produce(j);
}
```

You can handle items on the queue by overriding the consumer virtual method.

```
void consume(int & i)
{
    if (i%100 == 0)
    {
        std::cout << "Port = " << i << std::endl;
    }

    try
    {
        kbcafe::Socket socket;
        socket.Connect(name, i);
        std::cout << "Connected: " << name << " Port "
            << i << std::endl;
    }
    catch(kbcafe::SocketException &)
    {
    };
}
```

If the port scan utility is able to connect to a port, then it will respond that it Connected to the server on Port x. If the port connect fails, then nothing is reported. I report nothing on a failed connected as this is the normal behavior that I would expect and thus I'm only interested in deviations from this behavior.

I can then evaluate the result of the port scan and determine if I have any vulnerability. I wasn't able to connect to any ports inside our firewall, so I was very happy with the result. The only problem was that I enabled an alert to detect port scans and never once have I ever received the alert. I'll have to do a trace route someday to determine why the port scan is not being detected.