

Forum:[Hack Forums](#)

Thread: [Parallel Proxy Tester](#)

passwords - December 23, 2017, 9:45 am

Still needs work!

Sources

[https://msdn.microsoft.com/en-us/library/system.net.http.httpclienthandler.proxy\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.net.http.httpclienthandler.proxy(v=vs.110).aspx)

[https://msdn.microsoft.com/en-us/library/system.net.http.httpclient\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.net.http.httpclient(v=vs.110).aspx)

[https://msdn.microsoft.com/en-us/library/system.threading.tasks.parallel.foreach\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.threading.tasks.parallel.foreach(v=vs.110).aspx)

So I just threw together a quick proxy tester that works but it could work better.

The proxy tester only supports HTTP/HTTPS proxies as far as I know, if otherwise please correct me.

In a hurry so I wanted to make this post quick, might come back and update this

Source:

<https://github.com/Wolfijyn/SimpleParalellProxyTester/blob/master/ParallelProxyTester/Program.cs>

Source Clean (Not tested):

<https://github.com/Wolfijyn/SimpleParalellProxyTester/blob/master/CleanCode.cs>

Ninja - December 23, 2017, 9:56 am

Why would you run the loop in it's own thread? It's a Console Application, it won't freeze up. Also, you are not validating inputs, so if someone enters incorrect proxy strings the program crashes. And the Export function is useless, as you can do File.WriteAllText or File.WriteAllLines. And instead of Convert.ToInt32 I suggest you use "int.TryParse(string input, out int output)" as you get a bool returned whether or not the parsing was successful (this adds to the input validation bit). And instead of the if-statement on line 77 you could do this: "return Content.Contains("CmyIP.org");" to shorten it.

These are all minor things, and I don't mean to degrade you or your work as I love the release and effort you put up. This is meant for you to learn from.

PS: If you want, you can use sockets instead of the HttpClient to connect to the proxy server and check if it uses the protocol you want. This method is a bit harder but faster.

EDIT: This was for the first link. In the clean version you already fixed some of the issues I addressed, but not all of them.

passwords - December 23, 2017, 10:00 am

(Dec 23, 2017 10:56 AM)Ninja Wrote: [\[CITE\]](#) Why would you run the loop in it's own thread? It's a Console Application, it won't freeze up. Also, you are not validating inputs, so if someone enters incorrect proxy strings the program crashes. And the Export function is useless, as you can do File.WriteAllText or File.WriteAllLines. And instead of Convert.ToInt32 I suggest you use "int.TryParse(string input, out int output)" as you get a bool returned whether or not the parsing was successful (this adds to the input validation bit). And instead of the if-statement on line 77 you could do this: "return Content.Contains("CmyIP.org");" to shorten it.

These are all minor things, and I don't mean to degrade you or your work as I love the release and

effort you put up. This is meant for you to learn from.

PS: If you want, you can use sockets instead of the HttpClient to connect to the proxy server and check if it uses the protocol you want. This method is a bit harder but faster.

EDIT: This was for the first link. In the clean version you already fixed some of the issues I addressed, but not all of them.

I appreciate the feedback, and I mainly created this as a quick and dirty solution to test my proxies because I was looking for HTTP/HTTPS proxies specifically.

I'll rewrite the code later and I'll take your comment into consideration! ?

itcorp - December 23, 2017, 6:47 pm

(Dec 23, 2017 10:56 AM)Ninja Wrote: [\[CITE\]](#) Why would you run the loop in it's own thread? It's a Console Application, it won't freeze up. Also, you are not validating inputs, so if someone enters incorrect proxy strings the program crashes. And the Export function is useless, as you can do File.WriteAllText or File.WriteAllLines. And instead of Convert.ToInt32 I suggest you use "int.TryParse(string input, out int output)" as you get a bool returned whether or not the parsing was successful (this adds to the input validation bit). And instead of the if-statement on line 77 you could do this: "return Content.Contains("CmyIP.org");" to shorten it.

These are all minor things, and I don't mean to degrade you or your work as I love the release and effort you put up. This is meant for you to learn from.

PS: If you want, you can use sockets instead of the HttpClient to connect to the proxy server and check if it uses the protocol you want. This method is a bit harder but faster.

EDIT: This was for the first link. In the clean version you already fixed some of the issues I addressed, but not all of them.

I absolutely agree with you about usage of sockets is faster...

Here's a quick socket variant for test

```
Code public class ProxyList
{
    public string ProxyIP { get; set; }
    public int ProxyPort { get; set; }

    public ProxyList(string _proxyIP,int _proxyPort)
    {
        ProxyIP = _proxyIP;
        ProxyPort = _proxyPort;
    }
}

Code List<ProxyList> proxyList = new List<ProxyList>()
{
    new ProxyList("193.227.49.83",8080),
    new ProxyList("189.85.29.98",8080),
    new ProxyList("87.226.68.130",2020),
    new ProxyList("109.175.6.194",8080)

};

Code Parallel.ForEach(proxyList, pl =>
{
```

```

        ProxyTest(pl.ProxyIP,pl.ProxyPort);
    });Code private string ProxyTest(string ip,int port)
    {
        try
        {
            Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.SendTimeout,
2000);
            var _proxyIP = IPAddress.Parse(ip);
            var _endpoint = new IPEndPoint(_proxyIP, port);
            socket.Connect(_endpoint);
            if (socket.Connected)

                return $"Proxy {ip}:{port} is ok";
            socket.Close();
            return $"Proxy {ip}:{port} is faild test";

        }
        catch
        {
            return $"Error Proxy {ip}:{port} is faild test";

        }
    }
}

```

Ninja - December 23, 2017, 7:25 pm

(Dec 23, 2017 07:47 PM)itcorp Wrote: [CITE](#) - LARGE QUOTE -

Great suggestion, though I suggest input validation and more specific error handling so you know the exact reason a proxy doesn't work (incorrect proxy string, IP failed to parse, port out of range, host is down, host timed out, etc.). Also, instead of returning a string, you could return a bool and/or throw exceptions.

To OP:This method is not guaranteed, as that the host is up does not necessarily mean that it acts as a proxy; though the bad part is that you have to send an HTTP request to see if it indeed does act as a proxy. For socks proxies it is different and easier as there is a handshake and authentication protocol.

Here is my attempt [\(Click to View\)](#)

```

Code    private static void Connect(string host, int port, int timeout, bool ipv6 = false)
    {
        if (string.IsNullOrEmpty(host)) //You would assume that the Dns.GetHostAddresses throws
an exception on empty strings. It doesn't.
            throw new ArgumentNullException(nameof(host));

        IPEndPoint hostEntry = Dns.GetHostEntry(host);
        AddressFamily addressFamily = ipv6 ? AddressFamily.InterNetworkV6 :
AddressFamily.InterNetwork;
        IPAddress address = hostEntry.AddressList.FirstOrDefault(ip => ip.AddressFamily ==
addressFamily);
        if (address == default(IPAddress))
            throw new ArgumentException($"Invalid argument provided.", nameof(host));
    }
}

```

```

IPEndPoint endPoint = new IPEndPoint(address, port);
Socket socket = new Socket(addressFamily, SocketType.Stream, ProtocolType.Tcp);
IAsyncResult result = null;

try
{
    result = socket.BeginConnect(endPoint, null, null);
    result.AsyncWaitHandle.WaitOne(timeout, false);

    if (!result.IsCompleted)
        throw new TimeoutException($"The host did not respond in time.");

    socket.EndConnect(result);
}
catch (Exception ex)
{
    throw ex;
}
finally
{
    socket.Shutdown(SocketShutdown.Both);
    result.AsyncWaitHandle.Close();
    result.AsyncWaitHandle.Dispose();
    socket.Close();
    socket.Dispose();
}
}

```

This of course only checks if the host is up. If you want, you can keep open the socket instance and then try a few different protocols on that to identify which type of proxy it is (if it is one in the first place).

itcorp - December 24, 2017, 4:06 am

(Dec 23, 2017 08:25 PM)Ninja Wrote: [\[CITE\]](#) Great suggestion, though I suggest input validation and more specific error handling so you know the exact reason a proxy doesn't work (incorrect proxy string, IP failed to parse, port out of range, host is down, host timed out, etc.). Also, instead of returning a string, you could return a bool and/or throw exceptions.

Again agree with you. I gave this as an example your way is much more elegant and better from my i did not go deeper with this. Thanks for your reply

ForceJew - December 25, 2017, 12:54 am

Great job! haven't seen projects from you in some times, wasn't active;)