

Forum:[Hack Forums](#)

Thread: [ProxyGRB - Web Based Proxy Scraper](#)

w4zt3d - June 18, 2017, 12:18 am

I was able to pull 30K proxies with that

[Video: <https://www.youtube.com/watch?v=xl2B0nR0CXg>]

Theory

It's simple.

1. The tool searches for some queries like "proxylist" on Bing
2. Then it visits all the pages.
3. It grabs all proxies on each page using Regular Expressions (Regex).

Done!

NOTE: WE HAVEN'T CHECKED THE PROXYLIST YET!

Source

PHP Code

```
<?php
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Text.RegularExpressions;
```

```
using System.Net;
```

```
using System.IO;
```

```
namespace w4zt3d_ProxyGRB
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            string[] queries = { "fresh proxies", "pastebin proxies", "grayhatforum proxylist",  
"pastebin proxylist", ".txt proxylist", "pastebin proxy ip port" };
```

```
            Console.Title = "ProxyGRB by w4zt3d for GrayHatForum.org";
```

```
            int count = 15; // pages to scrape (all on first page npt looping this time)
```

```
            Regex linkRegex = new Regex("<a href=\\\"(.*)\\\" h=\\\"");
```

```
            Regex proxyRegex = new Regex(@"\d{1,3}(\.\d{1,3}){3}:\d{1,5}");
```

```
            Regex proxyRegex2 = new Regex(@"\d{1,3}(\.\d{1,3}){3}</td><td>\d{1,5}");
```

```
            WebClient webClient = new WebClient();
```

```
            List<string> alreadyFound = new List<string>();
```

```
            string filename = "scraped_proxylist.txt";
```

```
            File.WriteAllText(filename, "");
```

```
            int counter = 0;
```

```
            foreach (string tofind in queries)
```

```
            {
```

```
                string url = "https://www.bing.com/search?q=" + tofind + "\"&count=#count#";
```

```

try
{
    string code = webClient.DownloadString(url.Replace("#count#", count.ToString()));
    MatchCollection matches1 = linkRegex.Matches(code);
    foreach (Match match in matches1)
    {
        try
        {
            string s = match.Groups[1].Value;

            if (s.StartsWith("http") & !s.Contains("amp") & !s.Contains("microsoft"))
//amp & microsoft to avoid shit from Bing
            {
                string html = webClient.DownloadString(s);
                MatchCollection matches = proxyRegex.Matches(html);
                foreach (Match m in matches)
                {
                    string proxy = m.Groups[0].Value;
                    if (!alreadyFound.Contains(proxy))
                    {
                        counter++;
                        alreadyFound.Add(proxy);
                        Console.WriteLine(proxy);
                        File.AppendAllText(filename, proxy + Environment.NewLine);
                    }
                }
                matches = proxyRegex2.Matches(html);
                foreach (Match m in matches)
                {
                    string proxy = m.Groups[0].Value.Replace("</td><td>", ":");
                    if (!alreadyFound.Contains(proxy))
                    {
                        counter++;
                        alreadyFound.Add(proxy);
                        Console.WriteLine(proxy);
                        File.AppendAllText(filename, proxy + Environment.NewLine);
                    }
                }
                Console.Title = "ProxyGRB " + "\"" + tofind + "\" Proxys = " + counter + " Url: " +
s;
            }
        }
        catch { }
    }
}
catch { }
}
Console.Beep();
Console.WriteLine("Done!");
Console.Title = "ProxyGRB by w4zt3d " + counter + " proxies found.";
while (true) { }
}
}

```

}

Rageline - June 18, 2017, 12:35 am

Dont use regex to parse proxies. Its slow, it causes shit ton of memory allocation and GC fragmentation.

Also dont use foreach since it calls GetEnumerator on calls. Use for instead :)

Also why is there while(true) {} on end? U can just call Console.ReadLine(); and then program will exit when user presses enter or w/e

You should set a useragent because bing will block you for scraping

Also you should use HashSet<string> instead of List because it has O(1) lookup and can't contain duplicate items.

Also all collections in C# have a member called Count which returns number of elements in it. So you dont need your own counter variable

Ninja - June 18, 2017, 9:51 am

(Jun 18, 2017 12:18 AM)w4zt3d Wrote: [\[CITE\]](#) I was able to pull 30K proxies with that
This means nothing more than you put in a lot of sources or sources with lots of proxies. This does not mean anything more.

Anyways, as always the code is pure cancer.

SL x TnT - June 18, 2017, 2:36 pm

(Jun 18, 2017 12:35 AM)Rageline Wrote: [\[CITE\]](#) Dont use regex to parse proxies. Its slow, it causes shit ton of memory allocation and GC fragmentation.

I'm curious as to what you'd recommend for grabbing text from an unknown string with a known pattern. You'd also need an alternate solution before calling it slow.

Compared to using Contains on a List and http requests, Regex isn't slow.

Rageline - June 19, 2017, 2:55 am

(Jun 18, 2017 02:36 PM)SL x TnT Wrote: [\[CITE\]](#) I'm curious as to what you'd recommend for grabbing text from an unknown string with a known pattern. You'd also need an alternate solution before calling it slow.

Compared to using Contains on a List and http requests, Regex isn't slow.

Write your own grasshopper algo for proxy parsing, it takes ~ 30mins to write it. I'd share my own but im using it in commercial products so I don't want to.

Also OP your regex for proxies is wrong, u are capturing \d{1,5} for port which can capture all digits from 0-9 but ports are ushort size meaning 65535 is max and you will capture all invalid proxies above that :)

SL x TnT - June 19, 2017, 4:29 am

(Jun 19, 2017 02:55 AM)Rageline Wrote: [\[CITE\]](#) Write your own grasshopper algo for proxy parsing, it takes ~ 30mins to write it. I'd share my own but im using it in commercial products so I don't want to.

Also OP your regex for proxies is wrong, u are capturing \d{1,5} for port which can capture all digits from 0-9 but ports are ushort size meaning 65535 is max and you will capture all invalid proxies above that :)

Still going on about this "grasshopper algo," that's not really an algorithm, is well known, yet has no public information?

I'd rather spend the 10 seconds writing "slow" code than spend days searching for the elusive

grasshopper that will have no noticeable optimizations.

Optimizing everything from the start isn't the best route to go.

As for the port, it would be easier to check the value after you grab it.

+Haze - June 19, 2017, 6:36 am

(Jun 19, 2017 02:55 AM)Rageline Wrote: [\[CITE\]](#) Write your own grasshopper algo for proxy parsing, it takes ~ 30mins to write it. I'd share my own but im using it in commercial products so I don't want to.

Also OP your regex for proxies is wrong, u are capturing `\d{1,5}` for port which can capture all digits from 0-9 but ports are ushort size meaning 65535 is max and you will capture all invalid proxies above that :)

See, this is where you're going wrong, but you're too worried about insignificant optimisations. I've said this before and I'll say it again, premature optimisation is the root of all evil. Just like you said, it will take you ~30 minutes to code the algorithm for someone who understands how it works. Who wants to waste those 30 minutes?

In the context of modern computing, this performance difference is not noticeable, fair enough if this was the 90s where regex very slow for computers then, but this is 2017 son, it will make no difference to my i7 8 core laptop. Unless, the application is time sensitive then you're just shooting yourself in the foot with your non sense. I can understand if you were debating about using a quick sort algo vs bubble sort or an $O(\log n)$ vs $O(N^2)$ algorithm, but that's not the case.

Take a look at this

<https://softwareengineering.stackexchange.com/questions/80084/is-premature-optimization-really-the-root-of-all-evil>

Spoiler [\(Click to View\)](#)

So take a seat son.

(Jun 19, 2017 04:29 AM)SL x TnT Wrote: [\[CITE\]](#) x

I'd have to agree with you, and just for future references, the term is called premature optimisation.

HFDone - June 19, 2017, 7:16 am

(Jun 19, 2017 02:55 AM)Rageline Wrote: [\[CITE\]](#) Write your own grasshopper algo for proxy parsing, it takes ~ 30mins to write it. I'd share my own but im using it in commercial products so I don't want to.

Also OP your regex for proxies is wrong, u are capturing `\d{1,5}` for port which can capture all digits from 0-9 but ports are ushort size meaning 65535 is max and you will capture all invalid proxies above that :)

?

`\d` numeric digit 0-9
`{1,5}` 1 to 5 times

Rageline - June 19, 2017, 7:35 am

(Jun 19, 2017 07:16 AM)HFDone Wrote: [\[CITE\]](#) ?

`\d` numeric digit 0-9

{1,5} 1 to 5 times

5 times means 99999 which is not a valid port.

Ninja - June 19, 2017, 8:11 am

(Jun 19, 2017 07:35 AM)Rageline Wrote: [\[CITE\]](#) 5 times means 99999 which is not a valid port.
I love how you always ignore every single argument that you can't disprove. :)

Owl - June 19, 2017, 8:13 am

On the subject of regex, combine your delimiters into a charset/separate group and reduce capture work.

DBots - June 19, 2017, 8:19 am

Ever heard of [IPAddress.TryParse?](#)

HFDone - June 19, 2017, 10:50 am

(Jun 19, 2017 07:35 AM)Rageline Wrote: [\[CITE\]](#) 5 times means 99999 which is not a valid port.
??? you match not generate > 49999 is a valid port

SL x TnT - June 19, 2017, 1:10 pm

(Jun 19, 2017 06:36 AM)+Haze Wrote: [\[CITE\]](#) I'd have to agree with you, and just for future references, the term is called premature optimisation.
Thanks.

(Jun 19, 2017 07:35 AM)Rageline Wrote: [\[CITE\]](#) 5 times means 99999 which is not a valid port.
It doesn't matter if it's valid or not when finding the proxies. You should try to parse the IP as an IPAddress and the port as a ushort. If either fails, don't use the proxy.

Ninja - June 19, 2017, 2:26 pm

(Jun 19, 2017 04:29 AM)SL x TnT Wrote: [\[CITE\]](#) spend days searching for the elusive grasshopper
Gave me a good chuckle.

?