CountryBlocker

Akash Kapoor & Pranjit Kalita

Abstract

The modern data Internet user's data packets travel through various countries before reaching its destination. Some countries are involved in eavesdropping and tampering with the packets going through their routers. This calls for a tool that provides an explicit feedback to the user if the security or integrity of the transmitted data might have been affected because of the route. For our COS 561: Advanced Networks project, we are developing CountryBlocker - a tool that blocks web pages for which the data packet travels through a country blacklisted by the user.

1 Introduction

As part of our project, we developed a *Google Chrome browser extension* [5] - that would use tools such as *TraceRoute* [7] - to trace the route of the request. It detects if the packet was routed through a blacklisted country and blocks the user request if the it is expected to go through one of the blacklisted countries.

The extension was initially extremely slow and caused an uncomfortable lag, hampering the user experience. This was true even for the websites visited by the user frequently. To eliminate this, the extension makes intensive use of caching to make sure that the lag is minimal.

For the purposes of this project, we assume that the routes - especially the countries visited by the data packets - by the *TraceRoute* [7] call and when the actual page is loaded remains the same. Also, we assume that the IP address to Country mapping returned by *ip-api.net* [6] is accurate.

2 Design

CountryBlocker is a Google Chrome extension [5]. The design of the system has been shown in the next section (Flow Diagram) in two figures. In this section, we will highlight some of the major approaches we took towards creating the tool.

GUI for list of blocked countries : We provided a GUI to select a list of blocked countries, which would then be used to cross-reference with to decide which websites to block.

Traceroute API call via pro.viewdns.info : From the user provided URL and the HTML calls that are called on the backend, we extract each URL and make individual calls to a traceroute IP generating API, which we then parse to get the list of all IPs traversed. The API provides results in XML format.

IP->Country mapping via ip-api.net : For each extracted IP, we make a subsequent call to a IP-country mapping API, which produces an XML output. We parse that output to create a map structure between IPs and countries.

URL-Hopped Countries Cache : We maintain a cache for URLs and the list of countries for the IPs they hop to and call in turn (for example, to load images), to provide a faster solution just in case they happen to hop over one or more blocked countries. In case of cache hit at this level, we will iterate through the set of hopped countries to find a blocked one.

IP-Country Cache : Maintain a cache for IP-Country key-value pair so as to minimize time wastage in calling ip-api.net API. In case of a hit, we will immediately decide whether the website needs to be blocked. This cache is required explicitly differently from the URL cache since many websites could use the same sequence of initial hops (e.g. google.com and gmail.com), so many of the same IPs cached could come in handy to save time.

User Interface : If either through the two cache hits or simple API call returns any country from the list of blocked countries, then we will display an error message to the user stating that the website is blocked, and actually block the website. (See screenshots below)

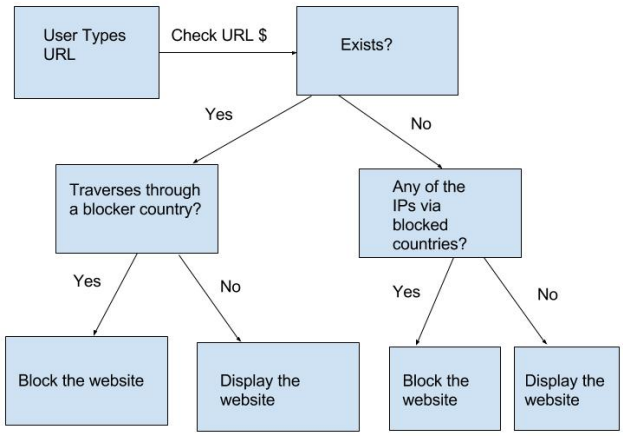

Figure 1: A general overview of the structure of country-blocker plug in in terms of functionalities.

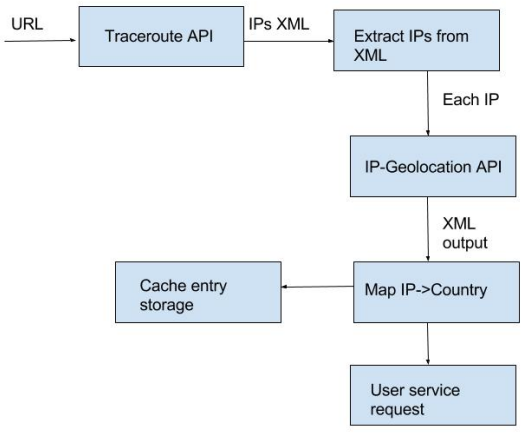

Figure 2: Sequence of calls to traceroute and ip-api.net to find appropriate IP->Country mappings.

In the above two figures, we show the basic outline of our plugin -

- Fig. 1 shows how our plugin is structured in terms of major functionalities, and how the execution trace is structured.
- Fig. 2 shows the interaction between our two API calls to calculate traceroute IPs, which in turn is used to calculate IP->country mapping, which are interfaced with the larger system in Figure 1.

2.1 User Interface

The extension provides user with a very simple graphical user interface to see the list of blocked countries, as shown in fig. 3. This list can helpful for the user to make sure all the offensive countries are being blocked.

There is also a mechanism in CountryBlocker to add or remove a country from the blocked list, as seen in fig. 6.

The extension provides a very visible feedback in form of notification with the offensive country name, as seen in fig. 4 and 5, to the user if the request page is blocked.

Our main objective behind this User Interface was to make it intuitive for the users to control the blocked countries, as well as make the user aware when a page was being blocked.

3 Evaluation

For our evaluation, we observed the cache behavior through logging and calculated the page loading times using a stop watch.

The main measure of our performance is through the usage of either of the two caches, which would in return not require API calls (to either traceroute or ip-api) in the background, which take a significant amount of time. In fact, calls to the traceroute API calls for uncommon websites (such as aljazeera.com, ndtv.com, etc) qualitatively take more than 30 seconds to load their corresponding XML files. The ip-api XML in comparison does not take nearly as much time, but caches could still improve their performances, as showin in the following data.

We compared the page loading time in the following 3 cases:

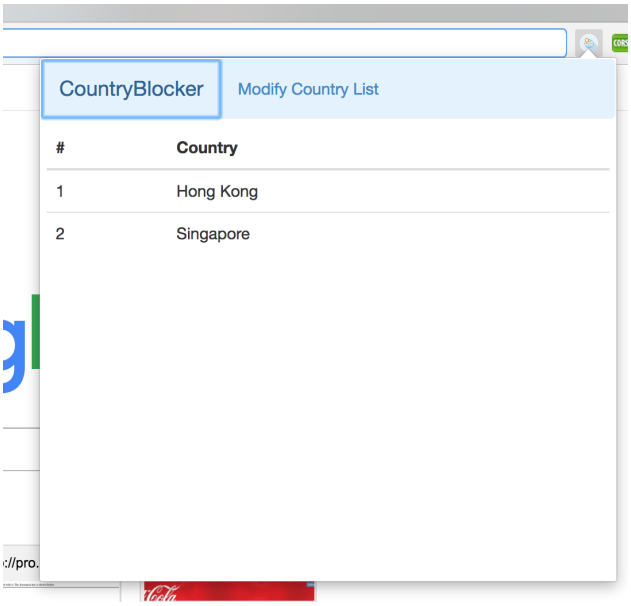

Figure 3: Interface to view blocked countries list.

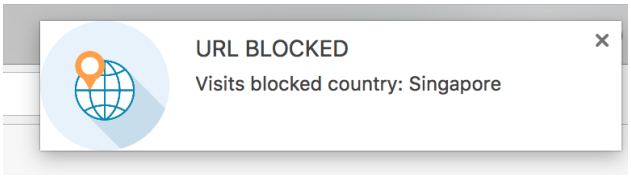

Figure 4: Notification - in case of a blocked page.

1. Cache Hit
2. Cache Miss
3. Extension Disabled

If our extension was disabled, loading Yahoo.com took approximately 4.51 seconds.

In case of a complete cache hit, as shown in Fig. 7, loading Yahoo.com took approximately 5.4 seconds. Thus, having an overhead of 19%.

On the other hand, a complete cache miss, as shown in Fig. 8, led to a loading time of 31.2 seconds. Translating to an overhead of 591%.

4 Related Work

A large amount of past work was instrumental in deciding course of the project. Some of the important related tangential work to inform our project included the following :

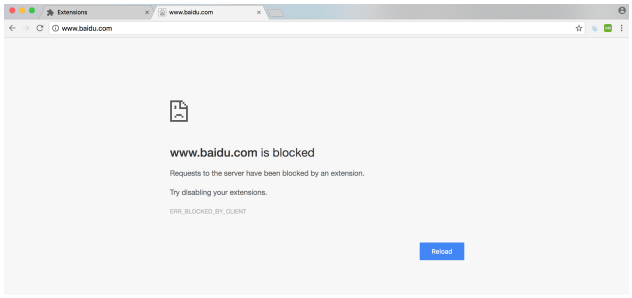

Figure 5: A blocked error page.

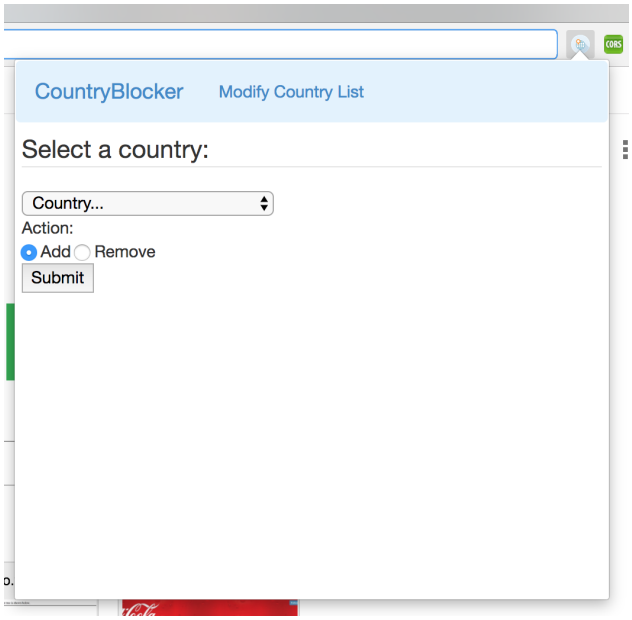

Figure 6: Interface to modify blocked country list.

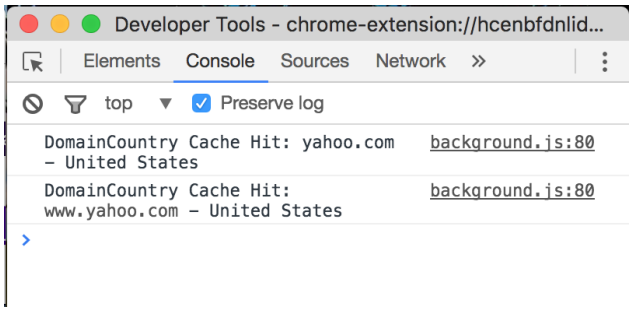

Figure 7: Logs in case of cache hit.

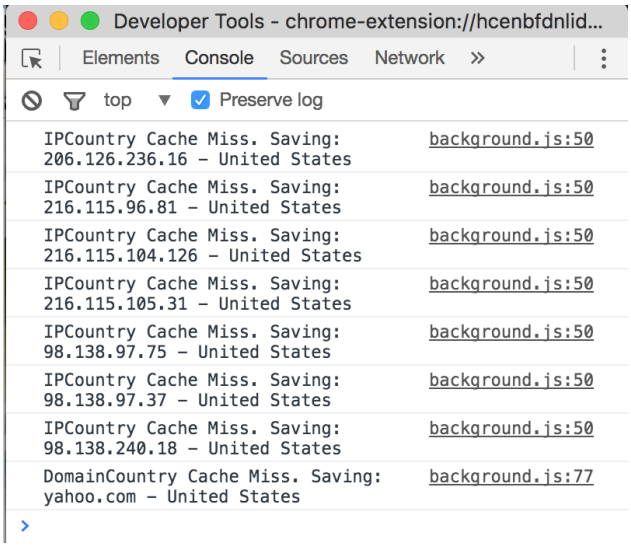

Figure 8: Logs in case of cache miss.

- Dealing with asymmetrical routes - We assumed the forward path from browser to the server to infer the reverse path. We looked into aspects of reverse lookup before settling on our assumption. [9, 10]

- We looked into a tool for calculating reverse traceroute based on IP Spoofing where the user did not have control of the destination, developed at the University of Washington. We decided to not use a similar tool because it would have been beyond the scope of the project. [2]

- IP to geolocation mapping through previous work done by Professor Nick Feamster, and his students. We specifically met with Annie Edmundson, a PhD student in Professor Feamster's lab, to gain insight into the tool (MaxMind) used to find the appropriate geolocation mapping from a given IP. Based on out discussion with her, we were led to believe that MaxMind is an internet registry of sorts of IP locations, which meant that the accuracy of the same could not be very well established. [1]

5 Conclusions

CountryBlocker is a tool to make user's online life safe and private. It allows users to block the traffic from going to a country that might be involved in eavesdropping or data tampering. The tool comes with a simple GUI that allows users to see their current blocked countries and modify the list. It makes an extensive use of caches to reduce the lag that user faces. With increasing hostility among countries and increasing threats to our online lives, it has become more important than ever to have a tool like CountryBlocker that provides us with means to protect the integrity and confidentiality of our data.

6 Acknowledgements

We like to thank Prof. Jennifer Rexford and COS 561 course staff for guiding us throughout the semester and giving their valuable feedback. We would also like to thank *Flaticon* [3], a website that let us download a free icon that we used for our Chrome Extension. The Javascript cache, *jscache* [4], by Monsur Hossain was extremely helpful and saved us a lot of time. Finally, we would like to thank *ip-api.net* [6] and *pro.viewdns.info* [8] for providing a free versions of their APIs.

References

[1] N. F. J. R. Anne Edmundson, Roya Ensafi. Characterizing and Avoiding Routing Detours Through Surveillance States, 2016. URL https://arxiv.org/abs/1605.07685v1.

[2] V. K. A. C. S. J. S. P. v. W. T. A. A. K. Ethan Katz-Bassett, Harsha V. Madhyastha. Reverse traceroute, 2010. URL http://www-bcf.usc.edu/~katzbass//papers/reverse_traceroute-nsdi10.pdf.

[3] FlatIcon.com. Free Vector Icons, 2017. URL http://www.flaticon.com.

[4] M. Hossain. Just a simple JavaScript LRU cache, 2017. URL https://github.com/monsur/jscache.

[5] A. Incorporation. Getting Started: building a chrome extension, 2017. URL https://developer.chrome.com/extensions/getstarted.

[6] IP-API. IP Geolocation API, 2017. URL http://ip-api.com/docs/.

[7] L. M. Pages. TraceRoute(8): linux man page, 2017. URL https://linux.die.net/man/8/traceroute.

[8] View-DNS.info. API-ViewDNS.info, 2017. URL http://viewdns.info/api/.

[9] J. R. R. K. Z. Morley Mao, Jia Wang. Towards an Accurate AS-Level Traceroute Tool, 2003. URL http://www.cs.princeton.edu/~jrex/papers/sigcomm03.pdf.

[10] J. W. J. R. R. K. Z. Morley Mao, David Johnson. Scalable and Accurate Identification of AS-Level Forwarding Paths, 2004. URL http://www.cs.princeton.edu/~jrex/papers/infocom04.pdf.