Poster: Cloud-Entry: Elusive Tor Entry Points in Cloud

Zhenlong Yuan*, Xiaoxian Chen‡, Yibo Xue*, Yingfei Dong†
* Tsinghua University, Beijing, China
† University of Hawaii, Honolulu, USA
‡ Harbin University of Science and Technology, Harbin, China
yuanzl11@mails.tsinghua.edu.cn, {chenxiaoxian, yiboxue}@tsinghua.edu.cn, yingfei@hawaii.edu

***Abstract*—As current Tor bridges and relays can be easily identified and blocked, it is critical for Tor users to be able to connect the Tor service. In this paper, we propose a novel *Cloud-Entry* service for Tor to address this issue by utilizing the unique characteristics of cloud services. As our first step, we have built a prototype pluggable transport, called *GTor* on the Google App Engine (GAE) platform, to help a Tor client to connect a Tor relay/bridge via a Google web application. This service is hard to identify and block, and thus also hides Tor users in cloud traffic. Furthermore, it makes identifying bridges a difficult task. Moreover, it naturally provides better performance with cloud resources. More importantly, it is scalable with the cloud service capacity. Our initial implementation and evaluations have demonstrated its effectiveness, and we are improving the scheme by developing cloud-based relays. We have made our GTor implementation available online.**

I. Introduction

As statistics-based methods can detect Tor encrypted sessions, and then find corresponding Tor relays/bridges, researchers have proposed to hide Tor traffic in other encrypted traffic in order to avoid being identified based on traffic characteristics. In particular, several protocol-level obfuscation plugins (i.e., *pluggable transport*) have been developed, such as SkypeMorph [1], StegoTorus [2] and CensorSpoofer [3]. Although they all achieved certain levels of obfuscation, a recent work [4] demonstrated that these pluggable transports failed to achieve completely unobservability. Even a weak local censor can easily distinguish their traffic from the imitated protocols. Therefore, new methods to protect Tor relays/bridges and hide Tor traffic are still open questions.

II. Proposed Cloud-Entry: GTor

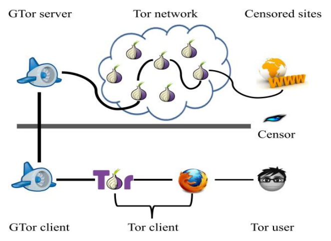
A. Unique Opportunities on GAE

GAE is a *platform as a service* (PaaS) platform for hosting web applications in Google data centers. For security and stability, all applications are sandboxed and run across multiple servers. Due to the scalability of the Google infrastructure, GAE offers automatic scaling for growing requests, i.e., allocating more resources for busy web applications to handle additional demands. The easy-to-use management of GAE eliminates the need for users to maintain their servers. They can simply upload their applications and then access them via their application IDs. GAE has a free service level with considerable storage and network bandwidth. Currently, one free Google account can support up to 10 applications, a maximum of 1GB free storage , and 1GB free bandwidth per day for each application.

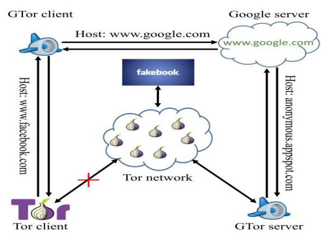
GAE supports several significant services, such as URL fetch and sockets. The URL fetch service allows GAE applications to exchange data with other hosts using HTTP requests and thus can help customers retrieve other web resources by using the high-speed Google infrastructure. The sockets service enables outbound sockets, and is available for billing-enabled applications for free. In summary, GAE is a powerful platform for running web applications. As a main product of Google cloud platform, GAE has a large number of high-capacity, high-bandwidth servers distributed all over the world. Note that the GeoIP databases show that the Google infrastructure has at least 730,000 IP addresses. GAE also supports automatic load balancing such that each of the above IP addresses can serve the same types of Google's services via HTTPS, e.g., search, Gmail, Gmap, and user-loaded applications.

Consequently, when we deploy web applications as the cloud-entry points for Tor on the GAE platform, we will enjoy the following benefits: (i) *Traffic analysis resistance.* Since every Google server can provide a wide range of services via encrypted SSL/TLS protocol concurrently, adversaries are unable to distinguish the types of services based on port numbers, IP addresses, or payloads. Statistic-based methods based on packet patterns are too costly for this purpose. (ii) *Service Availability.* In general, several Google services such as Google search and Google mail are indispensable for a region such that the adversary has to take a lot collateral damages if it simply blocks all Google servers (which is not a easy task anyway). Note that our GAE applications can work properly as long as there is a Google server available. (iii) *Secure entry-points publishing.* To resist the blocking of Tor non-public entry-points (i.e., bridges, obfs2 bridges and obfs3 bridges), the Tor project builds a few bridge pools and forces users to retrieve bridges' IP addresses by accessing the bridge management server via HTTPS or via a valid Google/Yahoo email account. However, although a few methods have been built for secure bridges publishing (e.g., Google reCAPTCHA has been used to prevent censors enumerating bridges from the HTTPS server), powerful censors still can collect bridge information by employing cheap labors. However, on GAE, the unique identification of a web application is its ID which is encrypted in a SSL/TLS flow. Even though a censor may gather such IDs, there is no practical way to match them in encrypted traffic.

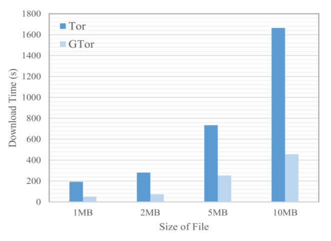
Most importantly, web applications on GAE are relatively easy to build, maintain, and scale without significant cost/effort



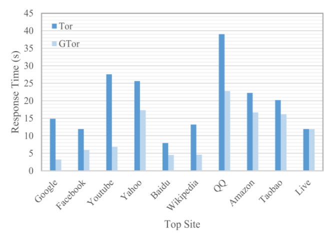
(a) Overview of the GTor Workflow.



(b) Strategy for Evading DNS Injection.



(c) Download Time Using "DownThemAll!"



(d) Response Time for Web Browsing.

Fig. 1. The Design and Evaluation of GTor

on the user side, as long as we control the total amount traffic at each application. So, we can set up a large number of Tor entry-points easily with almost no cost, which will certainly further improve the availability of Tor service in heavily censored regions.

B. GTor Design

As a *pluggable transport* of Tor, GTor sets up the first-hop connection of a Tor client to a Google web application, instead of Tor bridge/relay. Fig. 1(a) shows how GTor facilitates data exchange between a user browser and censored websites. GTor consists of a client-side plugin and a server-side cloud application. The client-side plugin is mainly responsible for wrapping a Tor flow into another further encrypted flow, and then sending it to the server-side application on GAE. The server-side application is mainly responsible for unwrapping the received encrypted flow and then sending it to a Tor bridge/relay as regular Tor traffic.

However, we have to overcome a few limitations of current GAE for web applications, especially when running in heavily censored countries. Some censors (such as the Great Firewall) use DNS injections to block users' access to censored websites [5], [6]. For example, the DNS responses in China for `appspot.com` are forged, which is the free domain for GAE users to run or connect to web applications. So, we have to solve this first.

Assuming that the application ID of a GTor server is `anonymous`, then the domain name would be `anonymous.appspot.com`. Users can contact this domain for the GTor service. However, the DNS response for `anonymous.appspot.com` in China will point to something else. To get around this DNS issue, we utilize the common Google search servers to deliver a HTTP request to a GTor server, as shown in Fig. 1(b). In particular, a GTor client first changes the "Host" field of a HTTP request to a Google search server from "`www.google.com`" to "`anonymous.appspot.com`", and then wraps another HTTP request (which originally should be directly sent to a GTor server) into the payload of the modified HTTP request. In this way, while receiving a special HTTP request whose "Host" field is "`*.appspot.com`", a Google search server will immediately forward the payload of the request to the corresponding web application. Note that the modified HTTP request is encrypted in a SSL/TLS flow when transferred from a GTor client to a Google server. So, censors can not detect this change easily. For the purpose of facilitating the further development and optimization on GTor architecture, we have made our GTor implementation (written in Python) open-source, available at: https://github.com/zlyuan/GTor

III. Performance Evaluation

In this section, we perform a preliminary evaluation by directly comparing the performance of GTor with Tor from Nov. 2013 to Jan. 2014. In our current implementation, we have implemented a GTor server (in a GAE web application) as an entry point to a real Tor relay or bridge, which actually adds one extra hop in a circuit in the cloud.

On a Windows PC, we use the reliable Firefox add-on "`DownThemAll!`" that supports downloading with resuming capability to evaluate the performance of GTor for downloading files, instead of using the TorPerf measurement tool supplied by Tor project. We downloaded 4 files with size 1MB, 2MB, 5MB, and 10MB, respectively, and repeated each download 100 times from the Wikipedia database. Fig. 1(c) shows the median download time for each file size. We can see that the average performance on GTor is much better than that on Tor, about 3.45 times faster on average, although the GTor added one extra hop in GAE in a circuit. To calculate the response times of the top 10 web sites from Alexa.com, we use the automated Firefox add-on "`iMacros`" that supports capturing precise web page response times with its built-in STOPWATCH command. Each web page was automated downloading 100 times and the median response times are shown in Fig. 1(d). We can see that the average response delay of web browsing with GTor is outperformed that with Tor, about 1.77 times faster on average.

References

[1] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM CCS*.

[2] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "Stegotorus: a camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM CCS*.

[3] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, "Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing," in *Proceedings of the 2012 ACM CCS*.

[4] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *Proceedings of the 2013 IEEE Symposium on S&P*.

[5] D. Anderson, "Splinternet behind the great firewall of china," *Queue*, vol. 10, no. 11, p. 40, 2012.

[6] P. Levis, "The collateral damage of internet censorship by dns injection," *ACM SIGCOMM CCR*, vol. 42, no. 3, 2012.