Circumventing Censorship with Collage

Sam Burnett, Nick Feamster, and Santosh Vempala
School of Computer Science, Georgia Tech
Atlanta, GA, USA
{sburnett, feamster, vempala}@cc.gatech.edu

ABSTRACT

Oppressive regimes and even democratic governments restrict Internet access. Existing anti-censorship systems often require users to connect through proxies, but these systems are relatively easy for a censor to discover and block. We explore a possible next step in the censorship arms race: rather than relying on a single system or set of proxies to circumvent censorship firewalls, we use the vast deployment of sites that host user-generated content to breach these firewalls. We have developed Collage, which allows users to exchange messages through hidden channels in sites that host user-generated content. To send a message, a user embeds it into cover traffic and posts the content on some site, where receivers retrieve this content. Collage makes it difficult for a censor to monitor or block these messages by exploiting the sheer number of sites where users can exchange messages and the variety of ways that a message can be hidden. We have built a censorship-resistant news reader using Collage that can retrieve content from behind a censorship firewall; we will show Collage's effectiveness with a live demonstration of its complete infrastructure.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—Security and Protection

General Terms

Design, Security

Keywords

anti-censorship

1. MOTIVATION

Network communication is subject to censorship and surveillance in many countries. The Open Net Initiative reports that 59 countries perform some degree of filtering [4], including more democratic countries such as the United Kingdom and Australia [3, 6, 7].

Although existing anti-censorship systems—notably, onion routing systems such as Tor [2]—have allowed citizens some access to censored information, these systems require users outside the censored regime to set up infrastructure: typically, they must establish and maintain proxies of some kind. The requirement for running fixed infrastructure outside the firewall imposes two limitations: (1) a censor can discover and block the infrastructure; (2) benevolent users outside the firewall must install and maintain it. As a result, these systems are somewhat easy for censors to monitor and block. For example, Tor has recently been blocked in China [5]. Although these systems may continue to enjoy widespread use, this recent turn of events does beg the question of whether there are fundamentally new approaches to advancing this arms race. We explore whether it is possible to circumvent censorship firewalls with infrastructure that is more pervasive, and that does not require individual users or organizations to maintain it.

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.
ACM 978-1-4503-0201-2/10/08.

We make a simple observation: countless sites allow users to upload content to sites that they do not maintain or own through a variety of media, ranging from photos to blog comments to videos. Leveraging the large number of sites that allow users to upload their own content potentially yields many small cracks in censorship firewalls, because there are many different types of media that users can upload, and many different sites where they can upload it. The sheer number of sites that users could use to exchange messages, and the many different ways they could hide content, makes it difficult for a censor to successfully monitor and block all of them.

We have designed Collage, a method for building message channels through censorship firewalls using user-generated content as a cover medium [1]. Hiding messages in photos, text, and video across a wide range of host sites makes it more difficult for censors to block all possible sources of censored content. Additionally, because the messages are hidden in other seemingly innocuous messages, Collage provides its users some level of deniability that they do not have in using existing systems (*e.g.*, accessing a Tor relay node immediately implicates the user that contacted the relay).

2. CHALLENGES

The first challenge we faced in designing Collage is developing an appropriate *message vector* for embedding messages in user-generated content. Our goal is to find user-generated traffic (*e.g.*, photos, blog comments) that can act as a cover medium, is widespread enough to make it difficult for censors to completely block and remove, yet is common enough to provide users some level of deniability when they download the cover traffic. We have built message vectors using Flickr and Twitter, but Collage's approach could be applied to any site with user-generated content and does not depend on these sites specifically. Although these services may ultimately be blocked in certain countries, a main strength of Collage's design is that blocking a specific site or set of sites does not top the flow of information through the firewall, since users can use many sites to post user-generated content.

Given that there are necessarily many places where one user might hide a message for another, the second challenge is to agree on *rendezvous* sites where a sender can leave a message for a receiver to retrieve. We aim to build this *message layer* in a way that the client's traffic looks innocuous, while still preventing the client from having to retrieve an unreasonable amount of unwanted content simply to recover the censored content. The basic idea behind rendezvous is to embed message segments in enough cover material so that it is difficult for the censor to block all segments, even if it joins the system as a user; and users can retrieve censored messages without introducing significant deviations in their traffic patterns. In Collage, senders and receivers agree on a common set of net-

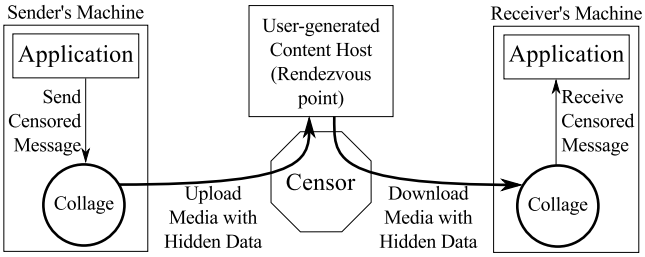

Figure 1: Collage's interaction with the network.

work locations where content should be hidden; these agreements are established and communicated as "tasks" that a user must perform to retrieve the content (*e.g.*, fetching a particular URL, searching for content with a particular keyword). Figure 1 summarizes this process. Users send a message with three steps: (1) divide a message into many erasure-encoded "blocks" that correspond to a task, (2) embed these blocks into user-generated content (*e.g.*, images), and (3) publish this content at user-generated content sites, which serve as rendezvous points between senders and receivers. Receivers then retrieve a subset of these blocks to recover the original message by performing one of these tasks.

3. ARCHITECTURE OVERVIEW

Collage follows a layered design that roughly mimics the layered design of the network protocol stack itself. There are three layers: the vector, message, and application layers. The *vector layer* provides storage for short data chunks and the *message layer* specifies a protocol for using the vector layer to send and receive messages. Many applications reside on top of the message layer.

The vector layer provides a substrate for storing short data chunks. Effectively, this layer defines the "cover media" that should be used for embedding a message. It hides the details of this choice from higher layers and exposes three operations: encode data into a vector, decode data from an encoded vector, and check for the presence of encoded data given a secret key. In our software release, we hide data inside JPEG images.

The message layer specifies a protocol for using the vector layer to send and receive arbitrarily long messages (*i.e.*, exceeding the capacity of a single vector). To send messages, users encode message data in vectors and publish them on content hosts; to receive messages, users find encoded vectors on content hosts and decode them to recover the original message. To distribute message data among several vectors, the protocol uses rateless erasure coding, which generates a supply of short chunks from a source message such that any appropriately-sized subset of those chunks can reassemble the original message. The sender takes chunks from the erasure coder, encrypts them and encodes them inside vectors using the vector layer.

Vectors are stored to and retrieved from user-generated content hosts; to exchange messages, senders and receivers must rendezvous by performing sequences of *tasks*, which are time-dependent sequences of actions. An example of a sender task is the sequence of HTTP requests (*i.e.*, actions) and fetch times corresponding to "Upload photos tagged with 'flowers' to Flickr"; a corresponding receiver task is "Search Flickr for photos tagged with 'flowers' and download the first 50 images." This scheme poses many challenges: (1) all tasks should resemble observable actions completed by innocuous entities not using Collage (*e.g.*, browsing the Web), (2) senders must identify vectors suitable for each task,

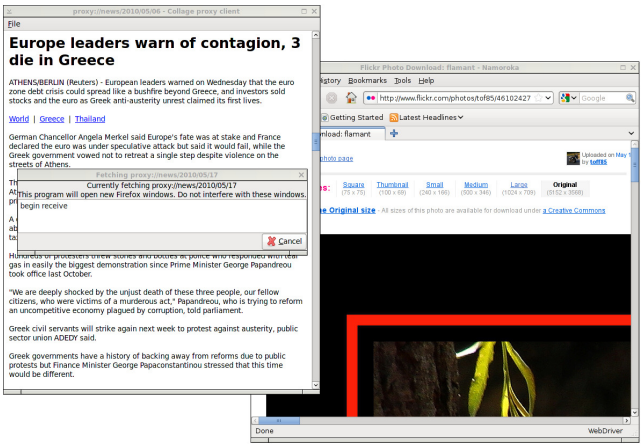

Figure 2: A news reader built using Collage. The window on the left drives the instance of Firefox shown on the right. The larger left window displays a downloaded news article.

and (3) senders and receivers must agree on which tasks to use for each message. Our full paper addresses these challenges [1].

4. IMPLEMENTATION

Collage requires minimal modification to existing infrastructure, so it is small and self-contained, yet modular enough to support many possible applications; this should facilitate adoption.

We have implemented Collage as a Python library, which handles the logic of the message layer. We have also implemented a variety of applications using Collage, including a news reader that extracts daily news feeds from images hosted on Flickr, shown in Figure 2.

To publish these news articles, the application needs a supply of cover images. We rely on benevolent contributors to user-generated content sites (*e.g.*, Flickr) to donate their photos to Collage. Before uploading these photos, the Collage donation service encodes each photo with censored content. To this end, we have implemented a donation service for Flickr and use it to collect photos, which we encode with our news articles.

Acknowledgments

This work was funded by NSF CAREER Award CNS-0643974, an IBM Faculty Award, and a Sloan Fellowship.

REFERENCES

[1] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *Proc. 19th USENIX Security Symposium*, Washington, DC, Aug. 2010.

[2] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.

[3] Uproar in Australia Over Plan to Block Web Sites, Dec. 2008. `http://www.nytimes.com/aponline/2008/12/26/technology/AP-TEC-Australia-Internet-Filter.html`.

[4] OpenNet Initiative. `http://www.opennet.net/`.

[5] Tor partially blocked in China, Sept. 2009. `https://blog.torproject.org/blog/tor-partially-blocked-china`.

[6] The Accidental Censor: UK ISP Blocks Wayback Machine, Jan. 2009. Ars Technica. `http://tinyurl.com/dk7mhl`.

[7] Wikipedia, Cleanfeed & Filtering, Dec. 2008. `http://www.nartv.org/2008/12/08/wikipedia-cleanfeed-filtering`.