1 Related work

Broadly speaking, there are two main challenges to proxy-based circumvention. The first is obfuscation: making circumvention traffic difficult to distinguish from traffic the censor wishes to allow. The second is proxy address unblockability: making it difficult for the censor simply to block proxies by address. Past works have considered one or both of these problems.

Traffic obfuscation has been approached in many different ways, which may be classified into two general techniques. The first technique is to look unlike anything forbidden by the censor; that is, fail to match a blacklist. The second is to resemble a protocol that is explicitly allowed by the censor; that is, match a whitelist. Falling into the first category are "look-like-nothing" transports whose payloads are indistinguishable from a uniformly random byte stream. The classic example of a look-like-nothing protocol is obfsproxy [10], long the go-to obfuscated transport used by Tor. ScrambleSuit [18] is like obfsproxy in the content of its payloads, but it takes additional steps to obscure its traffic signature (packet lengths and timing), and is designed to resist active scanning for proxies (the proxy server remains silent until the client proves knowledge of a shared secret).

The other category of obfuscation contains transports that take the steganographic approach: look like something the censor doesn't block. StegoTorus [15] encodes traffic to look like a cover protocol, such as unencrypted HTTP, using special-purpose encoders. Code Talker Tunnel (formerly called SkypeMorph) [11] mimics a Skype video call. FreeWave [8] encodes a digital stream into an acoustic signal and sends it over VoIP to a proxy which decodes and forwards it to the destination. Format-transforming encryption [3] encodes data into strings that match a given regular expression, in order to match a firewall's whitelist or avoid matching a blacklist.

Houmansadr et al. [7] evaluate "parrot" systems that imitate a particular implementation of a protocol and conclude that unobservability by imitation is a "fundamentally flawed approach." To fully mimic a complex and sometimes proprietary protocol like Skype is difficult in that the system must imitate not only the protocol's normal operation, but also its reaction to errors, its typical traffic patterns, and quirks of common implementations. Geddes et al. [6] demonstrate that even non-parrot systems may be vulnerable to attacks that disrupt covert communication while having little effect on legitimate traffic. Their examination is specific to VoIP protocols, where packet loss and duplication are acceptable. The censor may deliberately drop or inject ACKs in order to disrupt the covert channel, without causing much collateral damage.

The other grand challenge of proxy-based circumvention is proxy address unblockability, for which there have been a few approaches proposed. Tor has

long faced the problem of its entry relays being blocked. The list of relays is public, so it easy to block all of them by IP address. Tor bridges [2] are relays that are not universally known, intended to serve as entry points for censored users. A database of bridges (BridgeDB) seeks to provide a few secret bridges to anyone who asks, while at the same time making it difficult to learn the entire list. BridgeDB is capable of distributing the addresses of obfsproxy and ScrambleSuit bridges, and the combination of selective bridge distribution and traffic obfuscation has had some success as a comprehensive circumvention solution. It remains the case that there are simply not enough bridges to keep them all secret for long against a resourceful adversary.

Flash proxy [4] attempts to address the problem by conscripting web users as temporary proxies. Proxies last only as long as a web user stays on a page, so the pool of proxies is constantly changing and difficult to block. Flash proxy's approach to unblockability is in a sense the opposite of ours: where flash proxy uses cheap, disposable, individually blockable proxies, we use just one high-value proxy, which shares its fate with network infrastructure that is expensive to block. There exists a prototype transport that attempts to get both obfuscation and address unblockability by combining flash proxy with obfsproxy [12], however it is limited because it is not possible to obfuscate flash proxy's outermost WebSocket layer.

A technique known as OSS [5] (for "online scanning service") resembles ours in certain ways. OSS bounces data through third-party web services that are capable of making HTTP requests. Even though such services can fetch a web page, they do not in general preserve the contents of the page when it is returned to the requestor. (A translation service, for example, returns the page after translating it to another language.) For this reason, OSS does not rely on being able to send downstream data in HTTP response bodies, but rather requires the server to make a symmetric reflected HTTP request back to the client. The main ways in which this work differs is that we are effectively using an OSS that we fully control: we can ensure that reflected HTTP traffic is unmodified; and we make the OSS hard to block by hosting it on an important network resource.

Decoy routing is a recently proposed anti-blocking approach. Telex [19] asks friendly ISPs to deploy special software on routers between censored users and popular, uncensored Internet destinations. Circumvention traffic is marked with a special "tag" that is distinguishable from a random string only by Telex routers (not by the censor). On receiving such a tagged communication, the Telex router shunts it away from its apparent destination and toward the censored destination requested by the client. CensorSpoofer [14] decouples the upstream and downstream channels. Upstream data are carried over a low-bandwidth covert channel such as email. The CensorSpoofer proxy protects its address by spoofing the source IP address of all downstream data sent back to the client. In addition to spoofing, the CensorSpoofer authors propose to obfuscate the

downstream with a SIP-based steganographic channel. A recent study [9] on AS topology suggests that defeating decoy routing is likely to be expensive for the censors, if the decoy routers are strategically deployed. Despite that decoy routing is a sound technical approach, it is still questionable whether ISPs are willing to act against state-level censors. However, the takeaway is that the censors are unwilling to completely block day-to-day Internet access, which we can take advantage of. Of the decoy routing systems, Telex is the most similar to our work. Both systems use a piece of network infrastructure as a decoy to redirect certain flows (for Telex it is an ISP router; for us it is the Google frontend server), and both tag flows in a way that is visible only to the decoy router (for Telex a tag is a hash embedded in the TLS client randomness; for us it is the HTTP Host header).

Winter and Lindskog [17] investigated how China's Great Firewall blocks Tor. They confirmed an earlier discovery of Wilde [16] that the firewall identifies Tor relays using more than passive monitoring: it actively probes destination addresses to see if they speak the Tor protocol, and if so, blocks them for the future. (At the time, the firewall identified potential Tor connections for future probing by looking for a distinctive list of TLS ciphersuites—a failure of obfuscation.) The discovery of Chinese active probing was the motivation for probing resistance in ScrambleSuit and in our work.

GoAgent [1] is a direct inspiration for our system in its use of App Engine and Host header–based domain fronting. GoAgent requires users to upload a personal copy of the server code to App Engine, and works only with HTTP and HTTPS, not other TCP-based protocols. According to a May 2013 survey [13], GoAgent was the circumvention tool most used in China, with 35% of survey respondents having used it in the previous month. This figure is higher than that of paid (29%) and free VPNs (18%), and far above that of other special-purpose tools like Tor (2.9%) and Psiphon (2.5%). Users identified reliability, speed, and ease of installation as the most important features of a circumvention tool.

References

[1] GoAgent. `https://github.com/goagent/goagent`.

[2] Roger Dingledine and Nick Mathewson. Design of a blocking-resistant anonymity system. Technical Report 2006-1, The Tor Project, November 2006. `https://svn.torproject.org/svn/projects/design-paper/blocking.pdf`.

[3] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption.

In *Proceedings of the 20th ACM conference on Computer and Communications Security (CCS)*, November 2013.

[4] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Roger Dingledine, Phil Porras, and Dan Boneh. Evading censorship with browser-based proxies. In *Proceedings of the 12th Privacy Enhancing Technologies Symposium (PETS)*, July 2012.

[5] David Fifield, Gabi Nakibly, and Dan Boneh. OSS: Using online scanning services for censorship circumvention. In *Proceedings of the 13th Privacy Enhancing Technologies Symposium (PETS)*, July 2013.

[6] John Geddes, Max Schuchard, and Nicholas Hopper. Cover your ACKs: Pitfalls of covert channel censorship circumvention. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, November 2013.

[7] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The parrot is dead: Observing unobservable network communications. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (Oakland)*, May 2013.

[8] Amir Houmansadr, Thomas Riedl, Nikita Borisov, and Andrew Singer. I want my voice to be heard: IP over voice-over-IP for unobservable censorship circumvention. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS)*, February 2013.

[9] Amir Houmansadr, Edmund L. Wong, and Vitaly Shmatikov. No direction home: The true cost of routing around decoys. In *Proceedings of the 21th Annual Network and Distributed System Security Symposium (NDSS)*, February 2014.

[10] George Kadianakis and Nick Mathewson. Obfsproxy architecture. `https://www.torproject.org/projects/obfsproxy`, December 2011.

[11] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. SkypeMorph: Protocol obfuscation for Tor bridges. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, October 2012.

[12] Tor Project. #7167: Combine traffic obfuscation with address diversity of flash proxy. `https://trac.torproject.org/projects/tor/ticket/7167`, October 2012.

[13] David Robinson, Harlan Yu, and Anne An. Collateral freedom: A snapshot of Chinese users circumventing censorship. Technical report, Open Internet Tools Project, May 2013.

[14] Qiyan Wang, Xun Gong, Giang T. K. Nguyen, Amir Houmansadr, and Nikita Borisov. CensorSpoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In *Proceedings of the 19th*

ACM conference on Computer and Communications Security (CCS), October 2012.

[15] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. StegoTorus: a camouflage proxy for the Tor anonymity system. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, October 2012.

[16] Tim Wilde. Great Firewall Tor probing circa 09 DEC 2011. Technical report, January 2012. `https://gist.github.com/da3c7a9af01d74cd7de7`.

[17] Philipp Winter and Stefan Lindskog. How the Great Firewall of China is blocking Tor. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, August 2012.

[18] Philipp Winter, Tobias Pulls, and Juergen Fuss. ScrambleSuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, November 2013.

[19] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of the 20th USENIX Security Symposium*, August 2011.