DNS-sly: Avoiding Censorship through Network Complexity

Qurat-Ul-Ann Danyal Akbar
Northwestern University

Marcel Flores
Northwestern University

Aleksandar Kuzmanovic
Northwestern University

Abstract

We design *DNS-sly*, a counter-censorship system which enables a covert channel between a DNS client and server. To achieve covertness and deniability in the upstream direction, DNS-sly applies user personalization, adapting to individual behaviors. In the downstream direction, it utilizes CDN-related DNS responses to embed data, while retaining statistical covertness. We show DNS-sly achieves downstream throughput of up to 600 Bytes of raw hidden data per click on a *regular* Web page, making it a practical system in the context of a covert Web proxy service. We implement DNS-sly and evaluate it in a known censorship environment, demonstrating its real-world usability.

1 Introduction

Online censorship is thriving on the Internet [11]. Example scenarios range from governments blocking tens of thousands of Web sites [26], passing laws that make it illegal to digitally distribute content that opposes the government [26], openly denying service to targeted Web sites [8], to running large-scale firewalls that effectively isolate a country's Internet from the rest of the world [7].

To hide the content from a censor, users are often relying on encryption-based protocols and systems, *e.g.*, [2, 5, 9, 14, 15, 16, 19, 25, 29, 30]. Unfortunately, such systems fail to provide covertness and deniability for users. As a result, such attempts are easily discovered and blocked. For example, it has been reported that a censor is periodically resetting all encrypted connections automatically [26]. Hence, *designing systems and protocols capable of enabling censorship-free communication on the Internet is a vitally important task.*

We explore the potential of using the Domain Name System (DNS), the core Internet service [23], to build a large-scale counter-censorship system with strong covertness and deniability properties. The key premise of our work is that the significant network complexity in today's Internet, reflected in DNS, is an extremely useful asset in avoiding censorship. In particular, trillions of DNS requests are flooding the Internet every day [13]. Hence, any attempt to inspect, let alone analyze and control DNS traffic, is challenging. Next, the significant proliferation of public DNS services in recent years has largely blurred the notion of local DNS servers, forcing significant amounts of DNS traffic to flow through the Internet, crossing administrative country borders [13]. Finally, the substantial variability in user requests, accompanied by the diversity in DNS responses caused by the use of content delivery mechanisms, opens the door to significant information hiding opportunities.

Our system, called DNS-sly, illustrated in Figure 1, consists of requesters and responders communicating via DNS. A requester, running on a user's computer as a modified DNS client, and responder, running as a modified DNS server, communicate as a regular DNS client and a server. The DNS-sly responder provides a covert Web proxy service to the DNS-sly requester. To effectively circumvent censorship, it is essential that the DNS-sly presence at the client and the server is statistically undetectable. Our key design goals are the following:

- *Deniability for any DNS-sly requester*. It should be impossible to confirm that anyone is intentionally downloading information using DNS-sly, or to determine what that information is.
- *Statistical deniability*. A DNS-sly user's DNS request patterns should be statistically indistinguishable from those of regular DNS clients.
- *Responder covertness*. It should not be possible to detect that a DNS server is running DNS-sly by watching its behavior.
- *Resilience to attacks*. The DNS-sly system should be robust in the presence of censorship activities designed to interfere with DNS-sly communication.
- *Performance*. Despite all the above constraints, the

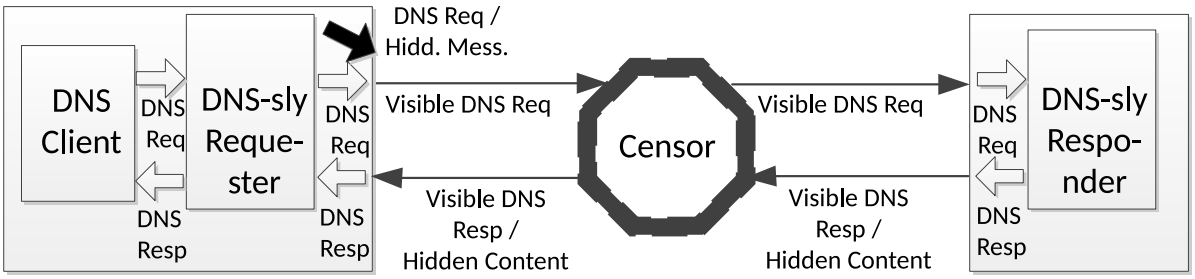

Figure 1: System Model.

system should enable performance useful in real-world censored environments.

To accomplish these goals, the DNS-sly responder first *learns* the DNS client properties, in terms of accessed domains and associated semantic topics, that the client is interested in. The system also collects the IP addresses related to requested domains, which are again user specific. Such knowledge about the user is then utilized by the requester and responder to effectively communicate while retaining covertness and deniability both for the DNS client and the server.

In the upstream direction, the DNS-sly requester crafts DNS requests (shown as a black arrow in Figure 1 on the left), which semantically overlap with the regular DNS requests, to ask for content from the responder. Upon receiving the request, the responder retrieves the content from the Web. Then, in the downstream direction, the DNS-sly responder encodes the desired content using DNS responses to *regular*, non-DNS-sly-requester-generated, DNS requests. This is done by encoding content in the *choice* of A records from the IP addresses of hosts associated with requested domains. We show that this approach not only enables communication in the downstream direction, but is also statistically indistinguishable from the regular DNS responses.

We make the following contributions:

- We demonstrate that network complexity, arising due to the significant variability in user behavior and the inherent inconsistency introduced by content delivery systems (reflected in DNS traffic), creates a rich content-hiding environment.
- We design DNS-sly, a system that utilizes network complexity to enable a counter-censorship service while providing strong covertness and deniability guarantees to participating users.
- We present a prototype which shows practical hidden-data throughput rates of up to 600 Bytes per click on a regular Web page.

This paper is structured as follows. Section 3 describes DNS-sly's design. Section 4 delivers DNS-sly's security and peformance evaluation. Section 5 discusses related issues. Section 6 mentions related work and Section 7 concludes.

2 Exploring Network Complexity

2.1 DNS Request Variability

Modern Web pages are typically fragmented, *i.e.*, pieces of content are often hosted at a number of different domains. Thus, requesting a single Web page means accessing numerous additional domains. This opens significant opportunities for DNS-sly. In the upstream direction, the large number of requests means that adding a small number of DNS-sly requests could be accomplished while retaining statistical deniability. In the downstream direction, the larger number of DNS responses increases the probability of DNS responses which are suitable for data hiding. We carried out an experiment using the Alexa Top 500 [1] and our findings show that the median is above 50 DNS resolutions per site, while around 20% of domains generated more than 100 resolutions.

2.2 DNS Response Variability

Here, we explore the number of IP addresses at which a domain is hosted. The larger the set, the greater potential for encoding our downstream data. We first explore the number of IP addresses at which a domain is hosted globally. This number provides an *upper bound* for the size of entries that we can use to design completely compliant DNS responses, *i.e.*, a count of responses in which each IP address actually points to a server which hosts the queried domain.

To estimate this value, we query the Alexa Top 500 domains, as well as all included domains on each page from 300 globally-distributed open DNS resolvers. Here, 25% of domains are served by a substantial number of IPs, likely due to the proliferation of CDNs. In particular, the median is at around 2,000 IPs, while the maximum is at nearly 16,000 IPs.

Although, DNS-sly can use the global set of IPs obtained to achieve reasonable downstream throughput,

this set doesn't provide the desired deniability. Therefore, we explore the number of IP addresses to which a domain resolves from a *single* DNS resolver. We refer to such sets of IP addresses per domain as *local*. These sets provide strong deniability properties, as they reflect addresses naturally seen at that location.

Our experimental results from 30 randomly-selected resolvers show that for the top 25% of domains the average number of IP addresses is smaller in comparison to the global set. The largest number of IP addresses per domain is approximately 850, about 19 times smaller than in the global case.

In addition to the IP set size, the number of A records in a DNS response further determines how much data can be embedded. We found that for the top 25% of domains, approximately one third of responses have more than 8 A records, and around 15% of responses have as many as 15 A records. This shows significant utility in the context of information embedding.

Timescales of DNS Response Changes If a DNS-sly server updates A records in DNS responses for a domain at the same rate at which they naturally change, it will blend accordingly. We first explain the simple metric we use to quantify a change in A records. Then, we measure this value in the real world.

We define *similarity* between two different DNS responses as the fraction of A records that have exactly the same IP addresses in the same position. Then, we define *change* as 1 - *similarity*. For example, consider three DNS responses: (IP1, IP2, IP3), (IP1, IP3, IP2), and (IP4, IP5, IP6). The change between the first and the second set of A records is $1-\frac{1}{3}=\frac{2}{3}$. The change between the first and the third set of A records is $1-\frac{0}{3}=1$. Given the *change* value between two DNS responses, we can consider the timescales at which they occur.

We query each of the top 25% of domains from the previous experiment once every 30 minutes and compute the change value between consecutive DNS responses. About 70% of domains completely change the location of their responses. It is well known that many CDN-hosted domains change such mappings, *i.e.*, at the order of tens of seconds [27]. This result clearly shows that the natural change in A records is high, enabling frequent reusability of the same domains to embed hidden data.

3 DNS-sly Design

Here, we present a detailed design of DNS-sly. The key segments of the DNS-sly protocol are: (*i*) end-point profiling, (*ii*) DNS-sly protocol bootsrapping, and (*iii*) communication

3.1 Endpoint Profiling

To retain covertness and deniability, DNS-sly adjusts its behavior relative to each individual DNS client. To achieve such personalization, the DNS-sly responder first profiles the clients DNS behavior. In the endpoint profiling phase, the DNS-sly responder records the DNS domains requested by the client. Furthermore, the DNS-sly responder collects unique IP addresses generated by an upstream DNS server for each domain.

The profiling phase is essential for covertness and deniability. In the upstream direction, the scope and diversity of the domains accessed by the client determine the properties of the upstream communication. Specifically, a set of requests carrying upstream communication are constructed from the endpoint profile to semantically overlap with the domains resolved in the profiling phase, maintaining covertness. In the downstream direction, the IP addresses associated with particular content replicas can often depend on the user's location in the network. This is due to the proliferation of the EDNS0 client-subnet extension [13]. The client-subnet allows a host issuing DNS requests to label requests with a subnet, indicating the origin of the request aiding in DNS-based replica selection, and in particular to address challenges which arise from clients being far away from their LDNS server. Hence, to retain covertness and deniability, it is necessary to log replica IP addresses selected for the client. At the end of the profiling phase, the responder creates a profile map of domains and corresponding IPs that it collected from quering an upstream DNS server.

3.2 DNS-sly Protocol Bootstrapping

Upon the creation of the user profile map, the DNS-sly responder needs a secure one-time out-of-band mechanism for distributing the map to the requester. As discussed in [12, 18], there are a number of ways to conduct one-time distribution without the knowledge of the censor, *e.g.*, clients could receive the profile map via postal mail or person-to-person exchange. Once the client receives the information, it can utilize DNS-sly's Web proxy services without any constraints.

3.3 Communication

Upstream communication comprises of DNS requests constructed using the user profile map. The content flows in the reverse direction, *i.e.*, from the DNS responder to the requester, embedded in regular, non-DNS-sly-initiated, DNS responses.

As ordinary DNS requests arrive from the requester, the responder watches for the occurrence of any domains which are contained in the profile map. When such a

domain arrives, the responder examines its entry in the map, noting how many IP addresses have been associated with it, *i.e.*, the size of its set of valid responses, which is denoted s and how many A records are in a typical DNS response for that domain, denoted c. Using this information, the responder is then able to compute the number of bits that can be encoded in this response. In particular, it computes the number of bits b as:

$$b = \lfloor log_2 \frac{s!}{(s-c)!} \rfloor.$$

That is, a choice of c A records from a potential pool of s IP addresses, can encode b bits. For implementation convenience, we then round this to the nearest byte, making our final message size $B = \lfloor \frac{b}{8} \rfloor$.

The responder then takes the first B bytes of the current page to send, denoted G_1. It then converts the integer value of the bytes of G_1 to their base-s equivalent. By controlling the length of G_1 above, we ensure that its representation would consume exactly c digits in its base-s representation. DNS-sly then considers the set of s IPs for that domain as an ordered list, allowing us to map each base-s digit to an IP address. The IPs corresponding to the digits of G_1 in base-s are then placed into the response in order and sent as a fully valid DNS response to the requester.

Upon receiving the DNS response, the requester first examines the entry in the user profile map that corresponds with the requested domain, determining the values of s and c. Using the IPs which appear in the A records, and their corresponding position in the ordered list of IPs for this domain, it is able to convert them to a representation in base-s, which is then converted back into their original binary representation. This data is then collected in a buffer until the entire message is received, with each chunk G_i sent in the response to each DNS request sent for a domain in the user profile map among the requester's organic traffic. Once complete, the message is decoded and delivered to the application. The requester and responder will not reuse the same domain to encode data until the Time-To-Live (TTL) from the authoritative server has expired. Since many of the domains in the user profile map are from CDNs, these TTLs may be as short as a few tens of seconds.

4 Evaluation

4.1 Security Evaluation

Discovery Attacks Joining DNS-sly, either as a requester or a responder, is challenging. Indeed, because the communication setup happens *out-of-band*, it assumes a level of trust between the requester and responder, upon which DNS-sly builds upon. While such an approach necessarily affects DNS-sly's scale, we opt for this approach because we give strict priority to deniability and covertness over scalability.

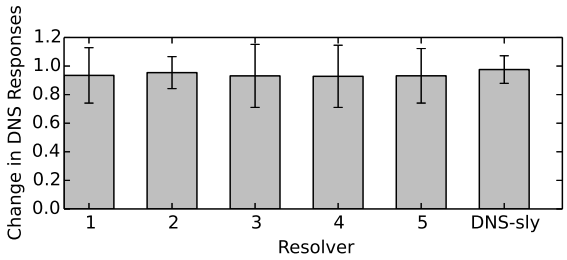

Figure 2: Change in DNS responses for DNS-sly responder compared to 5 other resolvers

Passive Attacks An adversary may try to discover potential DNS-sly participants or target a suspected DNS-sly requester by analyzing DNS traffic or inspecting an individual's request patterns respectively. However, all the DNS headers in DNS-sly are fully legitimate. Moreover, requests and responses by DNS-sly are indistinguishable from any other DNS packets in terms of packet sizes, entropy of hostnames, and record types. Upstream communication is hidden in regular user request patterns in terms of volume, frequency, and semantics. The covertness and deniability fundamentally arise from the profiling phase which enables a mapping personalized to each user. Thus, discovering DNS-sly is challenging even when DNS traffic is comprehensively examined.

Probing Attacks A censor can mount active discovery attacks, which we call probing attacks. In particular, an adversary can try to automatically detect a potential DNS-sly requester by probing other DNS servers, using the same or similar IP address as a suspected DNS-sly requester, at the same time.

We evaluate the probing attack as follows. For every response from a DNS-sly responder, the client also queries five other DNS resolvers for the same domain, thus emulating censor's probing attack. We conduct the experiment by querying all the domains we queried previously in Section 2 and compute the mean and variance of the change between each of the DNS resolvers and the other non-DNS-sly resolvers. The results in Figure 2 show exactly the same behavior of all the resolvers, landing well within the boundaries of a standard deviation.

Moreover, an attacker could attempt to force the DNS-sly responder to reveal itself by sending different DNS responses to the same DNS requests from the same DNS-sly requester over short time scales. DNS-sly protects against such attacks by explicitly prohibiting a DNS-sly responder from embedding information in a DNS response that has been issued within the TTL. Moreover, because the same DNS requests from the same user are not likely to happen in the real world, the performance

penalty for deploying this protection is minimal.

Filtering A censor may block access to certain content on the Internet. DNS-sly uses "regular" domains on the Internet that are hosted on many servers. If a censor sought to filter all such domains, it would effectively mean unplugging the Internet in a given area. While this is possible, *e.g.*, [3], the collateral damage of blocking all traffic is often too high [26]. A censor might further try to filter out DNS servers that support DNS-sly. To accomplish this, the censor would first have to discover them, which is challenging, as discussed above. In particular, the challenge comes from the out-of-band communication and the trust that exists between the requester and the responder.

DNS Tampering A censor might decide to tamper with the DNS responses by changing the order of IPs within DNS responses. However, without knowing the DNS-sly requesters and responders, it is hard to effectively scale this attack. As an example, Akamai's CDN serves trillions of client requests per day, controlling tens of terabits per second of content traffic served to clients world-wide [13]. Hence, inspecting and manipulating such a large volume of traffic is untenable.

Session Tampering A censor could serve a requester's visible DNS request from its own cache rather than forwarding this request to the DNS-sly responder. Such attacks are necessarily visible from the requester side. Moreover, without the ability to detect DNS-sly requesters and responders, the adversary would have to mount such an attack for *all* DNS requests, quickly reaching scalability limits.

4.2 Performance Evaluation

We built a prototype of DNS-sly in Python. The requester and responder are built to act as a DNS client and server. Compressed with bz2, the bootstrapping package, including the user profile map is 2.3MB. The requester can execute a series of DNS requests and decode messages which occur in a predefined user profile map. Since our client is designed for experimental purposes, rather than for use by live users, the upstream communications are hard-coded. While our implementation is a modest start at various levels, it allows us to evaluate the most important part of DNS-sly: its ability to push data in a censored environment.

Performance Here, we consider the downstream performance of the the system. In particular, we use a metric known as *bytes per click*. We define a single *click* to mean the loading of a page, including associated DNS lookups for both the original domain and all subsequent domains included on the page. Such a measure captures real user behavior, as the resolution of such names will always occur with regular user browsing.

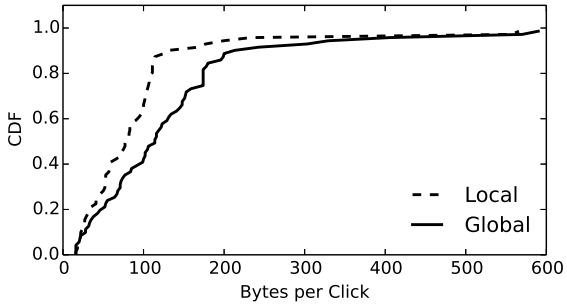

Figure 3: CDF of the downstream throughput from the global dictionary

Figure 3 shows a CDF of the number of bytes-per-click achieved for the local and global IP maps constructed with the top 15% of the Alexa Top 500 as described in Section 2. We see that the median page click with global provides DNS-sly with over 100 bytes of covert data and the local 75 bytes, only a 25% difference. The similarity of the local and global performance, despite their vastly different IP sets, is a function of the greater importance of the number of A-records, which is constant between the two maps. Furthermore, 30% of clicks offered over 150 bytes of data with the global map, with the largest observed site offering 600 bytes of data.

These findings place DNS-sly at the upper end of covert communication bandwidths. Indeed, measuring the size of DNS responses observed during this experiment, most responses feature of median size of 100 bytes, with 90% of responses under 200 bytes. Given such values, DNS-sly is comparable to other covert communication mechanisms [12].

We also carried out an actual 4KB file transfer, to observe bytes-per-click achieved. Our clicks were generated by selecting randomly from sites in the Alexa Top 500. We saw that, some clicks encode more information than others, still DNS-sly is able to take advantage of all of them. In total, the file completes its transfer in 30 clicks with the global profile map, and 64 with the local.

Behind a Firewall We further deploy DNS-sly in a known-censored environment. Specifically, the DNS-sly responder is run on a machine at our institution, and the requester is located in a country with known and active censorship. DNS-sly is able to successfully transfer a copy of a known-censored website, which we have reduced to text. It achieves the same rates observed in Figure 3 without a single packet loss, despite significant distance and delay between the requester and responder. The particular censorship environment is known to prohibit connections to censored sites, but in our experiments appeared to not interfere with DNS traffic at all.

5 Discussion

Unreliable Transport DNS-sly exists on top of the *unreliable* UDP protocol. While building a reliable communication on top of unreliable service is certainly possible [22], this potential design approach contradicts our key design goals, *i.e.*, covertness and deniability. Indeed, given that reliable protocols can exhibit deterministic and predictable behavior, they may make the presence of encoded content more obvious. Still, simple techniques, including the use of checksums and automatic re-transmits, could mitigate potential losses.

Impact on Regular Web Performance Given that DNS-sly encodes data by updating the A records originally set by content providers, one concern might be the effect that this could have on clients' perceived performance when accessing "regular" content. However, DNS-sly utilizes A records associated with servers that were recommended by the providers themselves, for a particular user. Hence, performance experienced by users when accessing such servers should still be good. DNS-sly could further minimize its effect on the performance for regular content by never changing the leading A record, used by default by most clients, but only the remaining ones.

DNS Caching DNS response caching is one of the primary services enabled by the DNS infrastructure, including the LDNS servers. The question is how DNS-sly interoperates with DNS caching. However, CDNs generally set DNS with very short timescales, to allow for DNS based replica selection to implement load balancing. Thus, in the absence of DNS-sly client activity, the DNS-sly-generated DNS responses quickly vanish.

6 Related Work

We survey other systems that aim to provide anonymous, confidential, or censorship-resistant communication. We draw distinctions between DNS-sly and such systems below.

DNS Tunneling A number of mechanisms have been developed in the past which tunnel IP traffic through a covert channel using DNS, often by encoding data in DNS payloads [17]. Examples of such systems include DNScat [10], DNScapy [28], iodine [4], TUNS [24] and many others [6, 21]. These systems differ in the record types they use for hiding data and the specific encoding technique used (*e.g.*, DNScat-B uses hex or NetBIOS encoding to hide content in A, CNAME, NS and TXT records and in [24] data is hidden within the TTL field). These techniques were not designed to be deniable and thus are easily detectable through DNS payload and traffic analysis [17]. DNS-sly provides much stronger deniabilty and is resilient to such analysis as discussed in our security evaluation.

HTTP Tunneling Systems related to DNS-sly in spirit but use HTTP tunneling for covert communication are Infranet [18] and Collage [12]. In the upstream direction, both Infanet and Collage encode hidden messages using sequences of visible HTTP traffic. However, in the downstream direction Infranet hides content in JPEG images using steganographic methods and Collage uses user-generated content published at various Web 2.0 sites to exchange hidden messages. The key difference between DNS-sly and Infranet is that Infranet enables a covert channel towards a single Web server, while DNS-sly does so towards a DNS server. This improves covertness, as an endpoint is far more likely to communicate to a DNS server than to any single Web server. Besides, Infranet uses standard JPEG steganographic methods, which are known to be prone to attacks [20]. While the use of streaming audio or video Web sites would be beneficial for Infranet, the choice of such specific Web sites might limit Infranet's scalability and availability. DNS-sly relies on the embedded dynamics in DNS responses induced by the significant proliferation of CDNs. Moreover, Infranet works in the context of the reliable TCP protocol, while DNS-sly operates on top of the unreliable UDP protocol, which leads to notable design differences. Unlike Collage, where a censor can blacklist an entire social network (e.g., Facebook in China), this is not feasible for DNS, which is a core Internet service.

7 Conclusions

We designed DNS-sly, a counter-censorship system that provides a covert channel between a DNS client and a server, with the following distinctive features: (*i*) DNS-sly adjusts its behavior to each individual user, and utilizes such knowledge to provide strong covertness and deniability guarantees in the upstream direction. (*ii*) It utilizes the frequently changing multiple A records in DNS responses in the downstream direction to enable equally strong guarantees for DNS-sly-enabled DNS servers, (*iii*) DNS-sly achieves downstream throughput of up to 600 Bytes of (uncompressed) hidden data per regular Web page click, making it a practically useful system. (*iv*) Our security evaluation showed that DNS-sly can successfully circumvent sophisticated censoring techniques, including active discovery probing attacks against DNS servers. (*v*) Finally, we demonstrated DNS-sly's utility in a real-censored environment, showing that it provides a novel, realistic, and reliable mechanism for avoiding censorship.

Acknowledgments

This project is supported by the National Science Foundation (NSF) via grant CNS-1526052.

References

[1] Alexa. http://www.alexa.com/.

[2] Anonymizer. http://www.anonymizer.com/.

[3] CBS News: How Egypt Pulled the Plug on the Internet. http://www.cbsnews.com/news/how-egypt-pulled-the-plug-on-the-internet/.

[4] iodine by kryo. http://code.kryo.se/iodine/.

[5] Squid web proxy cache. http://www.squid-cache.org/.

[6] tcp-over-dns. http://analogbit.com/software/tcp-over-dns/.

[7] The Great Firewall of China. http://www.greatfirewallofchina.org/.

[8] Ukrainian Authorities Suffer New Cyber Attacks. http://www.reuters.com/article/2014/03/08/us-ukraine-cricis-cyberattack-idUSBREA270FU20140308.

[9] Zero knowledge systems. freedom websecure. http://www.freedom.net/products/websecure/.

[10] BOWES, R. Dns cat, 2010.

[11] BURNETT, S., AND FEAMSTER, N. Encore: Lightweight measurement of web censorship with cross-origin requests. In *Proc. of ACM Sigcomm '15* (London, UK, 2015).

[12] BURNETT, S., FEAMSTER, N., AND VEMPALA, S. Chipping away at censorship firewalls with user-generated content. In *Proc. of USENIX Security '10* (2010).

[13] CHEN, F., SITARAMAN, R., AND TORRES, M. End-user mapping: Next generation request routing for content delivery. In *Proc. of ACM SIGCOMM '15* (London, UK, 2015).

[14] DANEZIS, G., DINGLEDINE, R., AND MATHEWSON, N. Mixminion: Design of a type iii anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (2003), SP '03, IEEE Computer Society.

[15] DINGLEDINE, R., FREEDMAN, M. J., AND MOLNAR, D. The free haven project: Distributed anonymous storage service. In *In Proc. of the Workshop on Design Issues in Anonymity and Unobservability* (2000), pp. 67–95.

[16] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13* (Berkeley, CA, USA, 2004), SSYM'04, USENIX Association, pp. 21–21.

[17] FARNHAM, G. Detecting dns tunneling. *InfoSec Reading Room* (2013).

[18] FEAMSTER, N., BALAZINSKA, M., HARFST, G., BALAKRISHNAN, H., AND KARGER, D. Infranet: Circumventing web censorship and surveillance. In *Proc. of the 11th USENIX Security Symposium* (Berkeley, CA, USA, 2002).

[19] FREEDMAN, M. J., AND MORRIS, R. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (2002), CCS '02, ACM.

[20] FRIDRICH, J., GOLJAN, M., AND HOGEA, D. Attacking the OutGuess.

[21] HOFFMAN, C., JOHNSON, D., YUAN, B., AND LUTZ, P. A covert channel in ttl field of dns packets. In *Proc. of SAM '12* (2012), p. 1.

[22] KUROSE, J. F., AND ROSS, K. W. *Computer Networking: A Top-Down Approach (6th Edition)*, 6th ed. Pearson, 2012.

[23] MOCKAPETRIS, P. *Domain Names - Implementation And Specification*. Internet Engineering Task Force, November 1987.

[24] NUSSBAUM, L., NEYRON, P., AND RICHARD, O. On robust covert channels inside dns. In *Emerging Challenges for Security, Privacy and Trust*. Springer, 2009, pp. 51–62.

[25] REITER, M. K., AND RUBIN, A. D. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur. 1*, 1 (Nov. 1998).

[26] SCHMIDT, E., AND COHEN, J. The Future of Internet Freedom. http://www.nytimes.com/2014/03/12/opinion/the-future-of-internet-freedom.html?&_r=0.

[27] SU, A.-J., CHOFFNES, D., KUZMANOVIC, A., AND BUSTAMANTE, F. Drafting behind Akamai (Travelocity-based detouring). In *Proc. of ACM SIGCOMM '06* (Pisa, Italy, Sept. 2006).

[28] VIXIE, P., AND WESSELS, D. Dnscapdns traffic capture utility. In *CAIDA Workshop, July* (2007).

[29] WALDMAN, M., AND MAZIÈRES, D. Tangler: A censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2001), CCS '01, ACM, pp. 126–135.

[30] WALDMAN, M., RUBIN, A. D., AND CRANOR, L. F. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9* (Berkeley, CA, USA, 2000), SSYM'00, USENIX Association, pp. 5–5.