Here Be Web Proxies

Nicholas Weaver[1], Christian Kreibich[2], Martin Dam[3], and Vern Paxson[4]

[1] ICSI / UC San Diego [2] ICSI / Lastline
[3] Aalborg University [4] ICSI / UC Berkeley

Abstract. HTTP proxies serve numerous roles, from performance enhancement to access control to network censorship, but often operate *stealthily* without explicitly indicating their presence to the communicating endpoints. In this paper we present an analysis of the evidence of proxying manifest in executions of the ICSI Netalyzr spanning 646,000 distinct IP addresses ("clients"). To identify proxies we employ a range of detectors at the transport and application layer, and report in detail on the extent to which they allow us to fingerprint and map proxies to their likely intended uses. We also analyze 17,000 clients that include a novel proxy location technique based on traceroutes of the responses to TCP connection establishment requests, which provides additional clues regarding the purpose of the identified web proxies. Overall, we see 14% of Netalyzr-analyzed clients with results that suggest the presence of web proxies.

1 Introduction

The World Wide Web continues to take center stage in people's use of the Internet. Indeed, for many users the web remains synonymous with the Internet itself. The plaintext nature of the web's workhorse protocol, HTTP, makes it particularly tempting to interpose on its flows using proxy servers, and HTTP remains one of the few protocols with explicit support for proxying. As a consequence, HTTP proxies have become widespread and different stakeholders employ them for a wide array of reasons. To the Internet's users, however, the actual prevalence and nature of web proxies remains a *terra incognita*. The typically transparent nature of web proxies means that users may remain unaware of their existence unless the proxy significantly malfunctions or induces significant changes to the connection payload.

In this work we present the results of extensive measurements probing for the presence of web proxies by conducting HTTP connections from end-user browsers to custom web servers under our control. We do so using Netalyzr [11], which contains a large and growing suite of proxy detection techniques. In 646,000 distinct addresses ("clients") analyzed by Netalyzr, 14% of clients show evidence of HTTP proxying

This work is supported by the National Science Foundation under grants CNS-0831535, CNS-1213157, and CNS-1223717, and the Department of Homeland Security (DHS) Science and Technology Directorate, Cyber Security Division (DHS S&T/CSD) Broad Agency Announcement 11-02, and SPAWAR Systems Center Pacific via contract number N66001-12-C-0128, with additional support from Amazon, Google and Comcast.

through one or more tests, suggesting that a significant fraction of all end-user HTTP traffic passes through web proxies either on the host or in the network.

We make two contributions. First, compared to the results we presented in the original 2010 Netalyzr paper [11], we now substantially broaden both the dataset (roughly seven times more sessions) and the depth of the analysis: we categorize the actual modifications, fingerprint proxy implementations, and, when feasible, deduce the purpose of the proxies' presence. Second, we introduce additional testing methods, including a proxy location technique based on traceroutes of the `SYN-ACK` packets responding to TCP connection requests. Given the improved measurement apparatus, we find nearly twice the fraction of HTTP-proxied sessions compared to our 2010 results.

We start by discussing the basic modes of operation in real-world proxies and presenting related work (Section 2). Next, we summarize Netalyzr's current proxy-detection test suite (Section 3), followed by a detailed presentation of our proxy fingerprinting and classification methodology (Section 4). We then present our findings (Section 5), including both identified proxies as well as a set of proxies whose purpose remains elusive, and a look at the most heavily proxied countries around the world. We conclude with a reflection on our findings (Section 6).

2 Background and Related Work

Web proxies examine and potentially alter some or all of a user's HTTP request and response traffic, sometimes even when the user has not explicitly configured the browser to route traffic through a particular proxy. In this work we consider both proxies co-located with the user's computer (such as security products) as well as in-path network elements.

Web proxies can employ two main strategies for modifying payload: TCP termination, and packet rewriting. A proxy employing TCP termination actively responds to the browser's TCP connection request, establishing a full transport connection with the browser, and creating a new, separate TCP connection with the target server. Once established, the proxy relays the content streams from both endpoints, potentially altering them at will. While we might expect the proxy to use its own IP address for the connection to the server, some proxies reuse the client's IP address. Doing so increases implementation complexity, but also provides transparency, and avoids any server-perceived centralization of behavior deemed abusive because it emanates in high volume from a single IP address.[1]

Packet-rewriting proxies, by contrast, modify traffic as it flows through them, potentially also injecting additional traffic, such as in the case of HTTP 404 error rewriting we observe in some NATs in Section 5. Packet-rewriting proxies work best for tasks that require only minor changes that can fit into a single packet, such as replacing a response entity with a redirection script.

[1] For example, BlueCoat's knowledge base (`https://kb.bluecoat.com/index?page=content&id=KB3119&actp=RSS`) specifically suggests enabling the "reflect-client-ip" configuration item (namely, use the client's IP address rather than the proxy's IP) in transparent mode, when Google detects a possible abuse situation. Operators can install such a proxy wherever symmetric routing ensures return traffic will transit it.

A substantial body of work covers Internet censorship detection [1,15,14], focusing on the general problem of triggering and understanding censorship mechanisms implemented using proxies or packet-injection tools.

Two academic studies have focused on specific proxy effects. The "Tripwires" work of Reis et al. [13] detected systems that modified HTTP content by performing an XML-RPC fetch and checking to see whether the returned content matched the expected content of the page itself. Huang et al. [10] used web ads in both Flash and Java to detect proxies based on flaws that make incorrect associations between hostnames and content to cache (per CERT VU 435052, as discussed below).

Finally, Auger proposed cache-detection using timing [2], where the origin server returns content after first observing an artificial delay. Objects that load quicker than the delay indicate the browser must be receiving the content from a caching proxy.

3 Detecting Web Proxies

In principle we can detect the presence of a proxy any time it permutes a connection's properties. We base our basic detection approach on employing an HTTP client and server under our control to exchange precisely known HTTP messages and then look for deviations from the expected. We implemented this approach using the ICSI Netalyzr, our popular user-driven, web-based connectivity analysis service that runs in a Java applet in the browser. See the original paper [11] for architectural and operational details, as well as general biases in our dataset, which remain largely unchanged. Netalyzr includes a range of tests that detect proxy implementation technologies, implementation artifacts and proxy limitations. Each user-initiated test session runs through a full suite of tests, of which we now describe in detail those relevant to HTTP proxy analysis. Since we have enhanced Netalyzr's test suite over time, we include for each of the test a description of the approximate number of distinct clients that observed the given results for the particular test.

Non-responsive Server Test (116,500 of clients tested): We expect TCP-terminating proxies, unless specifically customized, to respond with a `SYN-ACK` to a client's connection request before attempting to contact the client's intended origin server. We can test for this behavior by connecting to a server that we know will not accept the connection request [17]. For Netalyzr, we employ a server interface that sends a `RST` packet in response to all incoming requests, regardless of port. If the Netalyzr client's attempt to connect to this server on port 80 initially succeeds, this indicates the presence of a TCP-terminating proxy.

Proxy Traceroute (17,000 clients): The previous test indicates the presence of a TCP-terminating proxy but does not illuminate its location. We added to Netalyzr a new test to pinpoint the proxy's location, as follows. For any port on which the previous test flagged the presence of a proxy, the Netalyzr client attempts a TCP connection to our *traceroute server*. Upon receipt of an incoming SYN (likely sent by an in-path proxy), this server conducts a traceroute from server toward client using `SYN-ACK` packets. This traceroute terminates upon receiving the TCP handshake's pure `ACK`, rather than an `ICMP` "TTL exceeded" response. We do not perform a similar test outbound from the client, because while the client can technically invoke commands such as traceroute

directly, the issues of platform dependence, increased intrusiveness of the client, and the potential lack of required user privileges for a TCP-based traceroute make this approach problematic.

HTTP 404 Fetches (448,000 clients): While investigating DNS "error traffic monetization" [16], we discovered a proxy vendor whose product modifies HTTP 404 error responses. To detect this behavior, Netalyzr attempts to fetch three custom 404 "page not found" error pages. One returns just a blank 404 page, one returns a copy of Apache's default 404 page, and one returns Netalyzr's custom 404 page. We then watch for any alterations to the content.

Previously Documented Tests. In addition to the new tests described above, we used several existing Netalyzr tests in our analysis of web proxies.

Customized HTTP Fetch (633,000 clients): RFC 2616 [6] specifies that systems should treat HTTP header names as case-insensitive, and, with a few exceptions, free of ordering requirements. Netalyzr leverages these properties by implementing its own HTTP engine and fetching a custom page from the server, using mixed-cased request and response headers in a known order. Any changes indicate a proxy. This test also aids in the identification of the proxy's purpose. Some proxies declare their presence and/or function in a header, while others may modify the HTML document or transfer encoding in a manner which reflects the proxy's function, or serve as a base for further investigation (Section 4).

Non-HTTP Fetch (646,000 clients): In addition to a fetch using standard HTTP, Netalyzr attempts to fetch an entity using the protocol declaration `ICSI/1.1` instead of `HTTP/1.1`. A protocol-parsing proxy will likely reject this request as nonconformant.

Invalid Host Field (646,000 clients): Before Netalyzr's release, CERT VU 435052 [9] described how some in-path proxies would interpret the `Host` HTTP header and attempt to contact the listed host rather than forward the request to the intended address. We check for this vulnerability by fetching from our server with an alternate `Host` header of `www.google.com`.

Caching and Transcoding (619,000 clients): Netalyzr twice attempts to fetch an image URL from the server using a direct request that bypasses any local browser caching. Our server tracks first versus second requests, originally returning a particular 67kB image but for the second request returning an alternate version of the image. This process then repeats three more times, each time with different cache-control headers. If the client receives identical images for subsequent requests, we can deduce the presence of caching; altered images indicate transcoding. A more recent addition includes uploading the results of any transcoded images for further analysis.

Filetype Filtering (627,000 clients): Netalyzr attempts to fetch three different filetypes (`.mp3`, `.exe`, and `.torrent`), each representing a type of content that some network use policies may prohibit, and thus attempt to block with proxies.

EICAR Test Virus Filtering (296,000 clients): The initial Netalyzr release checked for the ability to receive the EICAR [5] test "virus," a benign program that antivirus programs recognize for testing purposes. We removed this test after receiving complaints about security software blocking all subsequent connections.

For similar reasons we do *not* include censorship-triggering tests. While technically straightforward to implement, we cannot rule out the possibility that such a test could result in harm to Netalyzr users who might be accused of accessing forbidden content.

4 Fingerprinting and Classifying Proxies

Using the Netalyzr results we just described, we set out to establish a methodology for fingerprinting the detected proxies and, in a subsequent stage, classify them into different categories of functionality.

Some of our tests naturally suggest a proxy's purpose, such as in the case of our caching analysis. We combined information gathered from our measurements with a manual, iterative rule-building approach in which we establish a set of detectors for specific proxy fingerprints. In each iteration, we identified the most prevalent proxy fingerprints and used them to infer the manufacturer and/or proxy model. Sometimes this task proved easy (such as when proxies inject a banner header, e.g., `X-BlueCoat-Via`); other times it required online searches and studying product whitepapers. Security and login gateways that block our requests generally present a page explaining their presence, while removal of whitespace suggests a transcoding proxy attempting to save bandwidth. Injected content or changes to the 404 error page also provide handy clues, as the injected URLs either directly disclose the company involved or help us track down the responsible parties in discussion forums or blogs.

In total, our resulting detectors comprise 70 generic rules for policy blockers (such as 'Blocked' or 'Denied' keywords) and 29 rules for individual content changes that alter received content.

5 Identified Proxies

Our analysis identified eight categories of web proxies. We sketch each in decreasing order of prevalence, and then discuss "dark proxies" that did not introduce any modifications that we could detect, and apparent country-wide proxies.

Antivirus (6% of clients): even though we removed the EICAR test due to collateral damage, triggering antivirus systems remains the most prevalent type of proxy for tested sessions. We also see indications of end-user security software through header changes validated by web searches. For example, Fortinet software uses a local proxy that adds an `X-FCCKV2` header to HTTP requests (210 clients). Note that we do not consider antivirus-blocking alone as an indicator of a proxy for other measurements; we only count sessions in which the HTTP connections exhibit evidence of proxying.

Caches (2.3% of clients): HTTP caches represent the second-most frequent proxy type. These systems attempt to reduce an ISP's upstream bandwidth by returning locally cached content instead of fetching it from origin servers. Since web clients possess their own cache, this only saves bandwidth on popular content.

Security and Censor Proxies (0.55% of clients): We detect two popular models of security proxies through the `Via` headers they inject. 1,156 clients indicate an IronPort/Cisco Web Security Appliance; 631 clients indicate a McAfee Web Gateway. Both proxies attempt to prevent attacks against web clients.

Similarly, the BlueCoat web filter, evident in 1,993 clients, can act as a security gateway (filtering dangerous content), an employee web-surfing censor, and/or a login gateway. This proxy inserts a `X-BlueCoat-Via` header in traffic to the server, while changes to the reply traffic consists of just a capitalization change in the `Connection` header and header reordering.

Finally, we received a session run by a volunteer behind a McAfee Smartfilter (operating as a censor) deployed in a Middle Eastern country. This proxy added a `Via: Webcat-Skein` request header and reordered and changed the capitalization on the `Connection`, `Host`, and `Cookie` headers, yet induced no reply header changes. We see 87 clients with this fingerprint.

Transcoding (0.54% of clients): While caches save upstream bandwidth, transcoding [7,8] conserves downstream bandwidth by compressing data into a more compact form. We observe three different transformations, usually applied in combination. The first consists of altered content encoding, replacing an uncompressed response with a gzip-compressed response. We observe 0.5% of clients that gzip-compress our HTTP 404 response or `.exe` file.

The second case, observed in 0.5% of tested clients, reflected proxies removing whitespace in the HTML content returned by our server. These transformations preserve HTML semantics (assuming that no HTML consumer relies on newlines in the rendering process). A common behavior is to compress the `.exe` file but newline-strip the HTML.

Finally, we also detect image transcoding that replaces our 67kB image with a smaller version. We observe 0.2% of tested clients with such modifications, usually preserving reasonable quality: most transcoding resulted in images greater than 22 kB (74.3%). The most compressed replacement consisted of a 5 kB image.

404 Rewriters (0.11% of clients): "Error traffic monetization" involves ISPs attempting to leverage protocol errors as a source of revenue by masking or augmenting the error delivery in order to include advertising [16]. While this controversial practice most commonly involves DNS NXDOMAIN errors, at least one company, Barefruit, also offers monetization of HTTP error traffic. This system requires the use of "a proxy device or DPI system to intercept returning HTTP errors" that the device replaces with a redirection to an advertisement-laden page.

We observe two ISPs, Mediacom (398 clients) and Bresnan(17 clients), that employ HTTP error monetization. The injected content looks identical except for the URL structure in the contained link, suggesting that both ISPs use a common provider for the 404-redirection, though the URLs differ in structure, which may reflect the ISPs working with different vendors for the landing page that offers up the ads. We also observed a bug in the injector: many sessions include the injected JavaScript snippet at the end of the response *headers*, as the injector did not insert an additional line break to separate the header from the injected body.

The DNS and WHOIS information for the Bresnan servers suggests that Xerocole operates the monetization, while Mediacom redirects to Infospace servers. In both cases these companies also provide DNS error monetization for these ISPs, suggesting either the ISPs or the monetization services use a common equipment vendor. Mediacom appears to have discontinued this technique after a public backlash in August 2012 [3].

According to our data, Bresnan appears to have never fully deployed this system, as we see indications of its use only among a small fraction of its users.

By leveraging Netalyzr's ability to query the local network for UPnP-enabled gateway devices and identify gateway device vendors [4], we can expand the analysis to proxying gateway devices. We observe that instances of the Linksys WRT110 contain 404-monetization (139 clients). This system redirects the user to `http://websearch.linksys.com` and does not appear to be part of the initial firmware, as devices with a manufacturer URL of `http://www.linksys.com` do not perform 404-monetization, while many (but not all) with a manufacturer URL of `http://www.linksysbycisco.com` do. This injector simply replaces the initial payload of the HTTP 404 response packet with a redirection, while keeping subsequent content intact. (Indeed, we observed a case where both Mediacom's injector (which injects a script but doesn't change the error code) and the Linksys injector operated on the same response!)

Login Gateways (0.075% of clients): Most login proxies operate within the private side of a NAT, enabling them to authorize connections based on a client's pre-NAT address. Login proxies that reside outside of a NAT, however, require some other means to track which clients the proxy has authorized. Some of these NAT-exterior proxies set a global "authorized" cookie for their own domain. When a new page request arrives from the browser, the login proxy first redirects to the authorization domain, checks for the cookie, and if present redirects the browser back to the original page, setting a cookie within the domain of the original page, whose presence flags subsequent requests to be passed through unmodified. Other configurations simply require that all requests go through a manually configured proxy. We observe 433 clients where our HTTP request encounters blocking by such a proxy.

Content Injectors (0.055% of clients): A comparatively rare class of proxies injects JavaScript or other content into HTML documents. The most common such injector, BitDefender (an antivirus solution seen for 318 clients), did not appear in the earlier survey of Reis et al. [13]. We also observed 58 clients containing an injection of "xpopup.js", part of the CA Personal Firewall popup-blocking suite running on client systems, and 11 clients showing evidence of Sunbelt Popup Killer, a dated (early-to-mid 2000s) anti-popup technology. Reis et al. likewise observed the latter two in their 2008 study [13]. Other injectors in our dataset include the privacy filter Privoxy, a VPN system by AnchorFree (here injecting advertisements), and Bluecoat and Comodo TrustConnect security products.

Three advertising injectors that operate on free hotspot connections appear in our dataset: Meraki toolbar,[2] Ovation Networks, and Icomera. These injectors insert a reference to a JavaScript routine that creates an advertisement-laden information bar on each page. We also observed an Indian ISP using Streamride to inject advertisements into all HTTP connections (injectors typically trigger only selectively [13]).

Some transcoding proxies also inject scripts. For example, we recorded a session by a vodafone.de customer that, in addition to stripping whitespace, injected a script `ups/ytchunk.js` into multiple pages. Web reports provide other mentions of such behavior, such as T-Mobile in the UK injecting a script `bmi-int-js/bmi.js`.

[2] Meraki's HTTP headers include: `X-cool-jobs-contact: jobs+proxyball@meraki.com`

Finally, we also observed an injector in SouthWest Airline's in-flight WiFi service. The injected toolbar conveys both flight information and branding, operating in a manner similar to the advertising injectors without third-party advertisements.

All of the above injectors can cause page loading/rendering problems. A common problem consists of injection-bearing web pages cached by the client that later can no longer retrieve uncached injected scripts (particularly those originally served from private or unallocated IP addresses). The host of Vodafone's injected scripts, for example, resides at 1.2.3.50, part of a reserved address block.

Spyware Proxies (0.036% clients): The OSSProxy.exe local proxy, part of the MarketScore software package (and considered spyware by Symantec) inserts an `X-OSSProxy` header with the software version.

Dark Proxies. For 8% of all clients in our dataset, we could detect the presence of a proxy via the non-responsive server test, changes to the HTTP headers, or diverging client addresses, but we could not identify any modifications due to the proxy. The first of these proved the most significant: prior to including a non-responsive server test in Netalyzr, 7% of clients show evidence of a dark proxy, while after adding the test this percentage rose to 12%. If we consider those measurements that included the non-responsive server test, but exclude measurements for which the proxy quickly responded to the initial `SYN` ($\leq$ 5 ms), the proportion of dark proxy clients is still 9%, indicating the likely presence of an in-network proxy on the far side of any NAT. Thus, the majority of these "dark proxies" reside internal to the network rather than at end systems.

Observations of dark proxies could reflect several possibilities. In settings with a login proxy (particularly hotspots), the user may have already authenticated to the login process prior to running Netalyzr, so the proxy at that point only relays content. Caches or transcoders that our tests do not trigger would likewise appear "dark" (although we expect to trigger these proxies as we provide numerous opportunities for them to cache and transcode data), as would proxies that enforce censorship or corporate policies that our probes did not trigger.[3] In a previous interaction with a Netalyzr user, we identified a workplace environment that uses proxies that only manifest in our measurements due to changes in the capitalization of the `Connection` header. In addition, we directly experienced one setting (the US National Science Foundation's internal network) that contains a proxy visible only via the non-responsive server and `SYN-ACK` traceroute tests, which locate the proxy in the same network. In both cases, the proxies' purpose was confirmed by visiting "forbidden" sites.

For the dark proxy clients where the connections arrived at the server from the same IP address as non-HTTP traffic, and for which we possess `SYN-ACK` traceroute results (1,345 clients), we attempted to determine the proxy's location via the traceroute data. We examined the last hop before the `ACK`-responding system in the traceroutes for both 80/tcp and for a non-standard port (1947/tcp). We considered these different if the last hop showed a different IP address; or, if one or both of the traceroutes failed to report the last hop, the hop count differed. Of the clients measured, 13% had a different traceroute

[3] For our purposes, whether such systems block "bad ideas" or "malicious content", from a network viewpoint they appear identical unless triggered.

for the two ports, suggesting that the proxy resides in the public network rather than at or inside any NAT.

Finally, regarding the possibility of dark proxies reflecting censorship proxies, we note some suggestive geography relating to the 197 clients that only manifested altered capitalization of the `Connection` header. When geolocalizing the IP addresses of these sessions (using MaxMind's GeoLite database), 113 resided in the United States (out of 147,000 total US clients), 20 in Kuwait (out of 223 clients), and 12 in Iran (out of 196 clients).

Country-level Proxies. Our data also show evidence in some cases of potential country-level proxying, where most or all of a nation's traffic passes through proxies. We examined all countries containing ≥ 50 distinct clients. Of these, the five with the highest prevalence of proxies are Bahrain (95%), Singapore (85%), Lebanon (79%), the United Arab Emirates (62%), and Thailand (48%).

Bahrain: While almost all Bahraini clients exhibited proxying, far fewer (42%) exhibited caching. The proxying very likely reflects censorship, as 86% of clients that successfully performed the customized HTTP fetch detected BlueCoat in the network, which has been previously linked to censorship in the Middle East [12].

Singapore: Again, a significantly lower percentage (26%) of clients manifest caching. Although we were unable to identify a common product by name, we noted that many Singapore clients reside behind a proxy that only adds an `X-Forwarded-For` header.

Lebanon: We suspect that the proxies we detect in Lebanon represent censors, as only 29% manifest caching. A common motif (51% of clients) is the addition of a `Cache-Control` header (even though almost no sessions actually exhibit caching), perhaps a `Via` header, and downcasing the `Connection` header on requests.

United Arab Emirates: Unlike the previous countries, the UAE manifests a higher degree of caching (41%), but 39% of clients also evince use of BlueCoat.

Thailand: Thailand shows a low degree of caching (17% of clients). There are also two somewhat common products, one (15% of clients) downcases all request headers, while the other (11%) adds a `Via` header.

Kenya: A single traceroute measurement to a client in Kenya indicates an apparent backbone-level cache. In the measurement session, an HTTP traceroute from our server to the client terminates after `82.178.159.110` rather than continuing to the next hop taken by a non-HTTP traceroute, `196.207.31.146`. AS-level information for these addresses indicates that they bridge the borders of Kenya and Oman.

6 Conclusion

Web proxies affect a significant fraction of Internet connections. Netalyzr's rich proxy-detection suite highlights proxies in 14% of the clients from which we have collected measurements to date—a significant increase from our 2010 result of 8%, which we attribute primarily to the significantly enhanced resolution of Netalyzr's proxy detection capabilities.

In addition to detecting the presence of proxies, we can often infer their of including caching, transcoding, login gateways, 404-rewriting, several types of content injection,

and local antivirus and spyware functionality. For those we cannot identify, we can still identify the network location as either at/within a NAT near the browser, or further upstream in the network. We can also detect and locate (but not classify) censorship proxies that terminate our HTTP connections.

At the country level, we find that Bahrain, Singapore, Lebanon, the United Arab Emirates, and Thailand all extensively manifest the use of proxies, with rates from 48% to 95%. Many of these do not appear to provide caching functionality, leaving nation-wide censorship as a likely explanation.

References

1. N. Aase, J. Crandall, A. Diaz, J. Knockel, J. O. Molinero, J. Saia, D. Wallach, and T. Zhu. Whiskey, Weed, and Wukan on the World Wide Web: On Measuring Censors' Resources and Motivations. In *Proc. USENIX FOCI*, Bellevue, WA, USA, August 2012.
2. R. Auger. Easy method for detecting caching proxies. http://www.cgisecurity.com/2011/02/easy-method-for-detecting-caching-proxies.html, February 2011.
3. CmdrTaco. Mediacom using DPI to Hijack Searches, 404 errors. http://yro.slashdot.org/story/11/04/27/137210/mediacom-using-dpi-to-hijack-searches-404-errors.
4. L. DiCioccio, R. Teixeira, M. May, and C. Kreibich. Probe and Pray: Using UPnP for Home Network Measurements. In *Proc. PAM*, Vienna, Austria, March 2012.
5. EICAR Anti-Malware Test File. http://www.eicar.org/86-0-Intended-use.html.
6. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, IETF, June 1999.
7. A. Fox, I. Goldberg, S. D. Gribble, D. C. Lee, A. Polito, and E. A. Brewer. Experience With Top Gun Wingman, A Proxy-Based Graphical Web Browser for the USR PalmPilot. In *Proc. Middleware*, 1998.
8. A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. Adapting to Network and Client Variability via On-Demand Dynamic Distillation. In *Proc. ASPLOS-VII*, October 1996.
9. R. Giobbi. CERT Vulnerability Note VU 435052: Intercepting proxy servers may incorrectly rely on HTTP headers to make connections, February 2009.
10. L.S. Huang, E.Y. Chen, A. Barth, E. Rescorla, and C. Jackson. Talking to yourself for fun and profit. *Proceedings of the Web 2.0 Security & Privacy (W2SP) Workshop*, 2011.
11. C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating The Edge Network. In *Proc. ACM IMC*, Melbourne, Australia, Nov. 2010.
12. Citizen Lab. Planet Blue Coat: Mapping Global Censorship and Surveillance Tools. https://citizenlab.org/2013/01/planet-blue-coat-mapping-global-censorship-and-surveillance-tools/.
13. C. Reis, S.D. Gribble, T. Kohno, and N.C. Weaver. Detecting In-Flight Page Changes with Web Tripwires. In *Proc. USENIX NSDI*, 2008.
14. A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis. Inferring Mechanics of Web Censorship Around the World. In *CensMon: A Web Censorship Monitor*, August 2011.
15. J. Verkamp and M. Gupta. Inferring Mechanics of Web Censorship Around the World. In *Proc. USENIX FOCI*, Bellevue, WA, USA, August 2012.
16. N. Weaver, C. Kreibich, and V. Paxson. Redirecting DNS for Ads and Profit. In *Proc. USENIX FOCI*, San Francisco, CA, USA, August 2011.
17. Wikipedia. Proxy server. http://en.wikipedia.org/wiki/Http_proxy#Detection, June 2012.