Int. J. Communications, Network and System Sciences, 2015, 8, 304-311
Published Online August 2015 in SciRes. http://www.scirp.org/journal/ijcns
http://dx.doi.org/10.4236/ijcns.2015.88030

Ultrasurf Traffic Classification: Detection and Prevention

Raed Al-Qura'n, Ali Hadi, Jalal Atoum, Malek Al-Zewairi

King Hussein Faculty of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan
Email: raedq@ju.edu.jo, ali@ashemery.com, atoum@psut.edu.jo, malek.alzewairi@computer.org

Received 24 June 2015; accepted 8 August 2015; published 11 August 2015

Abstract

Anti-censorship applications are becoming increasingly popular mean to circumvent Internet censorship, whether imposed by governments seeking to control the flow of information available to their citizens, or by parental figures wishing to shield their "parishioners" from the dangers of the Internet, or in organizations trying to restrict the Internet usage within their networking territory. Numerous applications are readily accessible for the average user to aid-in bypassing Internet censorship. Several technologies and techniques are associated with the formation of these applications, whereas, each of these applications deploys its unique mechanism to circumvent Internet censorship. Using anti-censorship applications in the work environment can have a negative impact on the network, leading to excessive degradation in the bandwidth, immense consumption of the Internet data usage capacity and possibly open the door for security breaches. Triumphing the war on anti-censorship applications has become more difficult to achieve at the network level due to the rapid updates and the adopted new technologies to circumvent censorship. In this study, a comprehensive overview on Internet censorship and anti-censorship applications is provided by analyzing Ultrasurf behavior, classifying its traffic patterns and proposing a behavioral-based solution that is capable of detecting and preventing the Ultrasurf traffic at the network level.

Keywords

Internet Censorship Circumvention, Proxy Avoidance, Ultrasurf

1. Introduction

Anti-censorship applications [1]-[5] aim to connect censored user to uncensored Internet. Ultrasurf is considered the most commonly used anti-censorship application [6]. Several technologies and techniques are employed to circumvent Internet censorship [7]. For instance, Ultrasurf employs a local proxy on the client machine, which listens on port 9666 to communicate with external proxies using Secure Socket Layer (SSL) on port 443. Additionally, it deploys different protection techniques to prevent dynamic analysis at run time, in addition to obfuscation techniques that are collectively known as binary packing as a method of preventing both static and dynamic binary analysis [8].

While anti-censorship applications can aid in securing freedom of speech; nonetheless, they can lead to excessive degradation in Internet bandwidth, immense consumption of Internet data usage capacity and possibly open the door for security breaches when used in the workplace.

For Small and Medium-sized Enterprises (SMEs) blocking anti-censorship traffic on the network level poses a great challenge, because traditional firewalls are incapable of inspecting data on the application layer; unlike Next Generation Firewalls (NGFWs), which impose the need to invest in new expensive technologies. Besides, anti-censorship applications (*i.e*. Ultrasurf) maintain and frequently update huge number of external proxies, meaning that it is not practical to block the external proxies IP addresses on perimeter firewalls. Additionally, the current NGFW technologies utilize signature based detection techniques, which make it costly in terms of time and human resources to keep up with their rapid updates.

In this paper, a behavior-based solution to detect and prevent Ultrasurf traffic at the network level is proposed. The researchers focus on Ultrasurf version 13.04 (MD5: E4C0EFE1C507F405D7B5DD169B842825, release date: 21-12-2013), study its behavior and acquire abnormal traffic from real and huge network, to be used in classifying Ultrasurf traffic. A Proof-of-Concept (PoC) software of the proposed solution was implemented using Python and Linux IP-Tables [9] to provide the needed routing functionality. The results show that the proposed solution is capable of detecting and blocking Ultrasurf traffic successfully with low false-positive ratio.

The rest of this paper is organized as follow: First, the latest research studies on Internet censorship and anti-censorship techniques are reviewed. Then, a brief overview on Ultrasurf is presented alongside with the proposed solution. After that, the PoC implementation used to evaluate the proposed solution is presented and the proposed solution and its limitations are discussed. Finally, the results of this study are concluded and the future work is outlined.

2. Literature Review

Network firewalls play a major role in providing fine-grained access control in both private and public networks [10]. Traditional firewalls, also known as stateless firewalls, are considered the most common type of firewalls, in which traffic is filtered based on sets of predefined rules constitute five features for both inbound and outbound packets, which are, source and destination IP addresses, source and destination port number and, fifthly, protocol. The Linux Netfilter (IP-Tables) is an example of host-based stateful firewall. Stateful firewalls inspect first outbound packet, then all related subsequent packets are allowed in both directions (inbound and outbound) [10]. Unlike traditional firewalls, which are unaware of the content of the traffic; therefore, incapable of filtering traffic based on its content (application layer) or blocking anti-censorship applications, NGFWs are preferable for controlling network traffic in enterprises; however, are still considered an expensive solution for SMEs.

Houmansadr *et al.* proposed an Internet censorship evading system "Cirripede" that works at the Internet Service Provider (ISP) level with the ability to camouflage anti-censorship traffic to appear as if it is coming from innocent destinations using routing deflection techniques in addition to asymmetric cryptography. Their system aims to languish repressive regimes ability to detect and identify individuals who use anti-censorship technologies [11].

Wang *et al.* propose "CensorSpoofer", a web browsing anti-censorship framework that implements asymmetric connection approach consist of direct and indirect connection channels. The indirect channel sends concealed request of censored Websites using steganography within legitimate traffic such as emails to uncensored proxies. Whilst, the direct connection channel delivers censored content over covert channels such as VoIP while hiding the proxy's real IP address from both users and censors using IP spoofing techniques [12]. Nonetheless, [13] [14] demonstrate that both active and passive adversary can easily detect and block covert channels used by CensorSpoofer without having any negative impact on legitimate VoIP traffic due to having differences between legitimate VoIP protocol and its censorship circumvention mimic.

Weinberg *et al.* addressed protocol analysis on Tor traffic issue and propose "StegoTorus" framework to enhance Tor resistance against fingerprinting attacks. Their solution employs pluggable cryptographic-based steganography with two modules: StegoTorus-Embed module that imitates peer-to-peer traffic such as Skype and VoIP. Secondly, StegoTorus-HTTP module that mimics HTTP client/server request/response to conceal Tor traffic between Tor client and Tor relay server using SOCKS proxy [15]. However, Houmansadr *et al.* argue that both steganography modules proposed by StegoTorus can be distinguished with ease from legitimate traffic using passive attacks [13].

Burnett & Feamster examined the various challenges that censorship monitoring systems face and briefly suggested several measurements to help achieve better understanding of why a particular content is being censored and what exactly triggers it [16].

Jones *et al.* proposed an automated method for censorship monitoring capable of detecting returned "block pages", in addition to fingerprinting five products commonly used in Internet censorship. Three similarity measures were studied, which are page length, cosine similarity using HTML tags as term frequency vectors and DOM Similarity. The results indicated that using page length with threshold of 30% produced the highest Precision rate of 79.80% [17].

Anderson explained the different techniques employed by the government of China to censor the Internet through the Great Firewall of China (GFC). Moreover, technical and economic implications caused by the GFC were discussed [18].

Khattak *et al.* presented an analysis study of two nationwide censorship campaigns in Pakistan dated in 2011 and 2012, which included the blocking of both porn content and YouTube respectively. The results showed that in both events a significant increase in encrypted traffic occurred shortly after the imposition of the censorship. Additionally, a notable drop in the use of the local DNS resolvers arose, indicating that different anti-censorship techniques were employed [19].

Chaabane *et al.* provided a unique study on the Internet censorship in Syria by analyzing log files leaked from seven proxies used in nationwide censorship from a nine days period between July and August 2011 [20]. The study identifies four metrics utilized in censorship that are consistent with similar filtering activities. However, unlike other nationwide censorship campaigns, traffic filtering in Syria tends to be more targeted for specific media as the study shows prevalent censorship on instance messaging applications, entire Israeli subnets and topics on the Syrian Revolution [21].

3. Proposed Solution

Ultrasurf is a single hop client/server proxy system that is popular between users seeking to circumvent Internet censorship as it provides easy-to-use experience, portability (no installation is required) and excellent reliability in bypassing censorship. Ultrasurf client is a Microsoft Windows only application that creates a local proxy on the user machine listening, by default, on port 9666 at the loopback IP address (127.0.0.1), and then connects to one of Ultrasurf external proxies. It relies on a non-standard SSL/TLS protocol on port 443 to encrypt the traffic between the Ultrasurf client and the proxy server [8]. Ultrasurf received multiple praises for its role in securing freedom of speech [22] [23].

Appelbaum provided a comprehensive technical analysis of Ultrasurf architecture (both client software and network) claiming that they have multiple security and design flaws that could undermine its purpose [8]. Nonetheless, UltraReach published a detailed response acknowledging some of the point raised by Appelbaum, whilst, refuting the majority of others [24].

Osman *et al.* addressed the issue of blocking Ultrasurf traffic in SMEs [25]. They propose using Squid cache proxy server [26] to block all TCP connections on port 80 and 443 that request a host directly by its IP address.

3.1. Ultrasurf Traffic Analysis

In this study, the researchers examined traffic generated from Ultrasurf client (version 13.04) and used Wireshark to interpret and analysis the captured packets. The analysis environment consisted of two machines, one is running Ultrasurf client on a Windows 7 (x64) and the other is running Wireshark on an Ubuntu 12.04 (x64). Both machines were connected to the same access switch. However, the Ultrasurf machine is connected to the Internet through a proxy server, whilst the Wireshark machine is connected to a span port that is mirroring both egress and ingress traffic of the Ultrasurf machine. Analyzing the captured traffic showed that Ultrasurf client, usually, begins its connection by sending a DNS query requesting Google domain name using the machine default DNS server. It then initiate a secure TCP connection (SSL/TLS) to one of Ultrasurf external proxy serveries on port 443.

In order to understand how Ultrasurf client behaves under DNS hijack censorship, a modification in the testing environment were made to block all DNS query responses to the Ultrasurf machine. In this case, Ultrasurf client started sending DNS query requests for Google, Appspot and Amazon respectively within a maximum period of 45 seconds as follow: in case the DNS query requesting Google (google.*) was blocked, it tried requesting Appspot (appspot.com) and in case that was also blocked, it tries requesting Amazon (s3-ap-southeast-1.amazonaws.com). Finally, if all failed, it will resort to connecting directly to one of the hardcoded Ultrasurf proxy servers. We assume that Ultrasurf client is designed to update its list of IP addresses for the Ultrasurf proxy servers by calling cloud services such as Google Drive, Google Cloud Platform, and Amazon Simple Storage Service. The analysis also showed that the Ultrasurf client invokes domain-fluxing attack whenever it suspects that domain hijacking might be deployed by censors. **Figure 1** shows the architecture of the analysis environment explaining the Ultrasurf client connectivity flow.

3.2. Classifying Ultrasurf Traffic

In order to detect Ultrasurf traffic, we propose to exploit its client behavior rather than blocking the IP addresses of its proxy servers, because as explained previously, this list is dynamically updated with each run of Ultrasurf client. The proposed solution works by delaying DNS query requests of Google, Appspot, and Amazon for *n* seconds, where *n* value is a variable, thus forcing Ultrasurf client to try to connect to one of its hardcoded proxy servers via TCP on port 443.

Additionally, a reputation table, in which each client in the network has a value ranging from one to three, is maintained. This value is increased if the client made a DNS query request to one of the three domain names

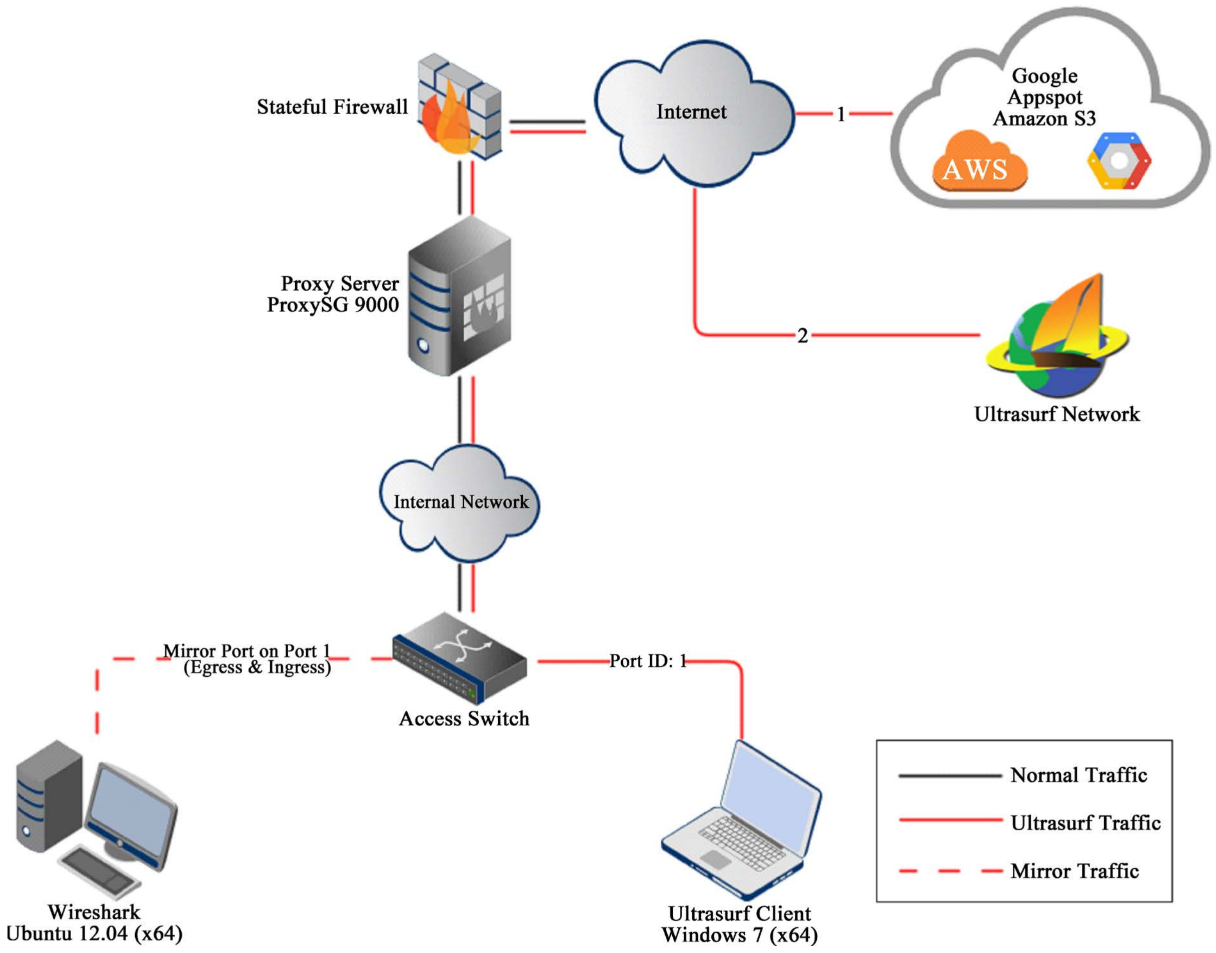
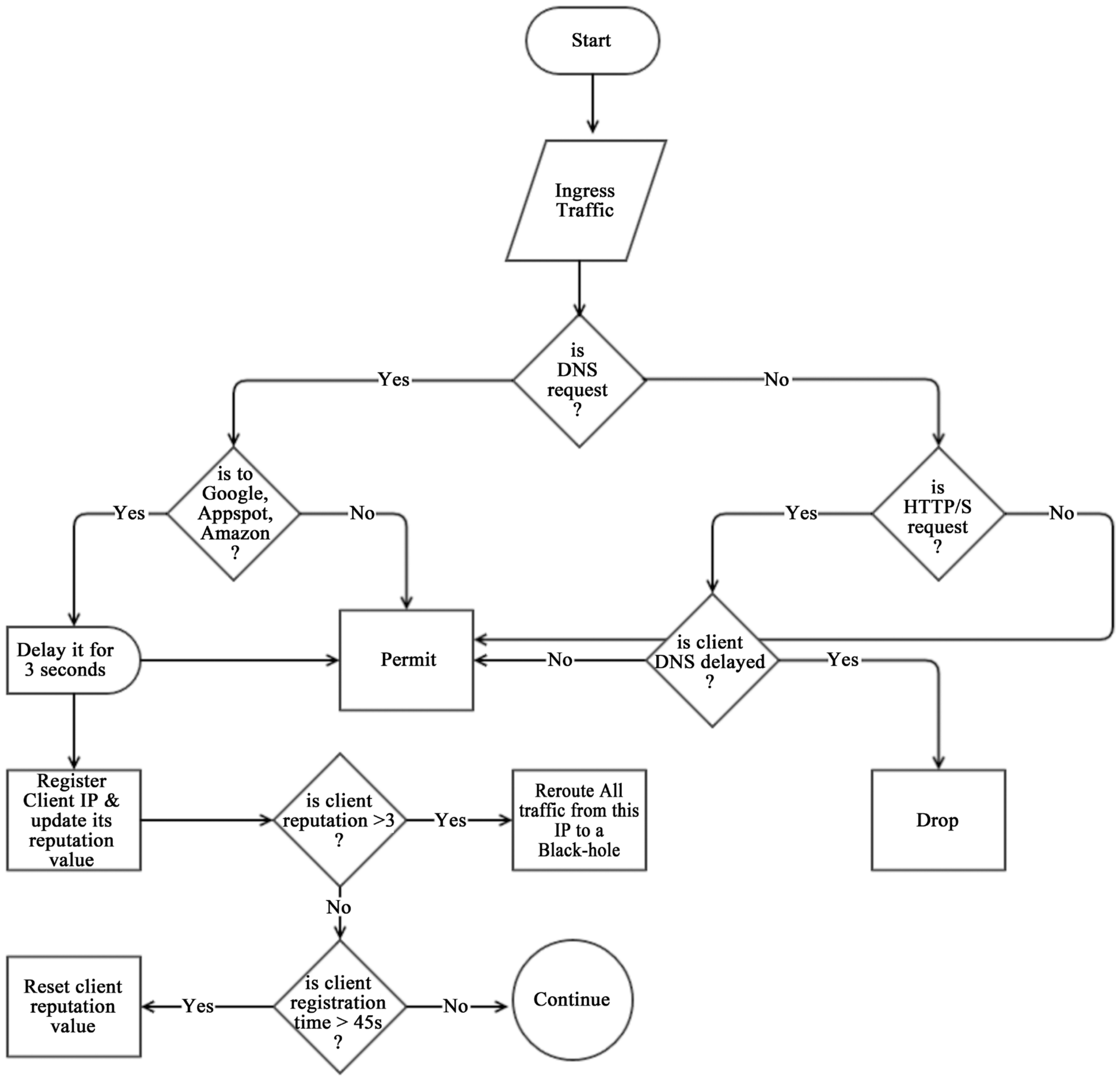


Figure 1. Architecture of Ultrasurf client connectivity flow.

(*i.e.*, Google, Appspot, and Amazon) within *n* seconds. Meaning that, in case a network node requests all three domains within *n* seconds, its reputation will be set to the maximum value. Meanwhile, all TCP connection made on ports 80 and 443 within *n* seconds is dropped, whenever the reputation value of the client is less than the maximum value. In case, the client's reputation value is equal to the maximum value, this is used as an indication that the client is most likely running Ultrasurf, and thus rerouting its traffic to a black hole. Otherwise, all traffic is permitted. A reset of the client's reputation value is made every 45 seconds. **Figure 2** presents the decision-making flowchart of the proposed solution.

4. Evaluation and Discussion

In order to evaluate the proposed solution, a PoC of the proposed solution was implemented using Python to detect and block Ultrasurf traffic. The implementation environment consists of three virtual machines, two of which are running Windows 7 (x64) and the other is running Ubuntu 12.04 (x64). The Ubuntu machine acts as a gateway for the other two machines, for which one connects to the Internet via Ultrasurf client (v13.04), while the other imitates normal Internet user behavior.



Start
Ingress Traffic
is DNS request ?
Yes
No
is to Google, Appspot, Amazon ?
is HTTP/S request ?
Yes
No
Yes
No
Delay it for 3 seconds
Permit
is client DNS delayed ?
No
Yes
Register Client IP & update its reputation value
is client reputation >3 ?
Yes
Reroute All traffic from this IP to a Black-hole
Drop
No
Reset client reputation value
Yes
is client registration time > 45s ?
No
Continue

Figure 2. Decision-making flowchart of the proposed solution.

The implemented PoC takes advantage of Linux IP-Tables to provide packet filtering and packet forwarding tasks. Additionally, it utilizes NFQueue [27] to transfer issuing the verdict on a packet from the kernel-space to the user-space; thus, allowing the application to inspect the packet and issue a verdict in accordance with the proposed design. A stateless packet inspection is used, where all incoming packets are accepted, however, the decision on outgoing DNS request queries, HTTP and HTTPS packets (UDP on port 53, TCP on ports 80 and 443) is delegated to the Python application.

As explained previously, DNS query requests to Google, Appspot, and Amazon are delayed for *n* seconds, in the implementation a 3 seconds delay period was adequate to ensure a detection ratio of 71.82% without imposing high latency. Moreover, a PowerShell script was implemented to automate the evaluation process by letting the Ultrasurf client execute for two minutes, in order to simulate an attempt to bypass the proposed solution. We repeat this step for 100 times each one with different *n* value ranging from one to ten seconds. The log file of the Python application was then analyzed and the average successful detection ratio for each different value of *n* was calculated.

Figure 3 presents the architecture of the implementation environment. The source code of both the Python application and the PowerShell script is published at
https://github.com/alzewairi/ultrasurf_traffic_classification_detection_and_prevention.

Table 1 shows the different values of DNS delay period (*i.e.*, *n* seconds) and their expected detection ratios resulted from analyzing the PowerShell script results. Nonetheless, we noticed that increasing the delay period consequently enhances the detection ratio, yet obviously increases the latency. It is worth mentioning that the detection has significantly improved because of enhancing the Python application code to reduce the number of I/O operations.

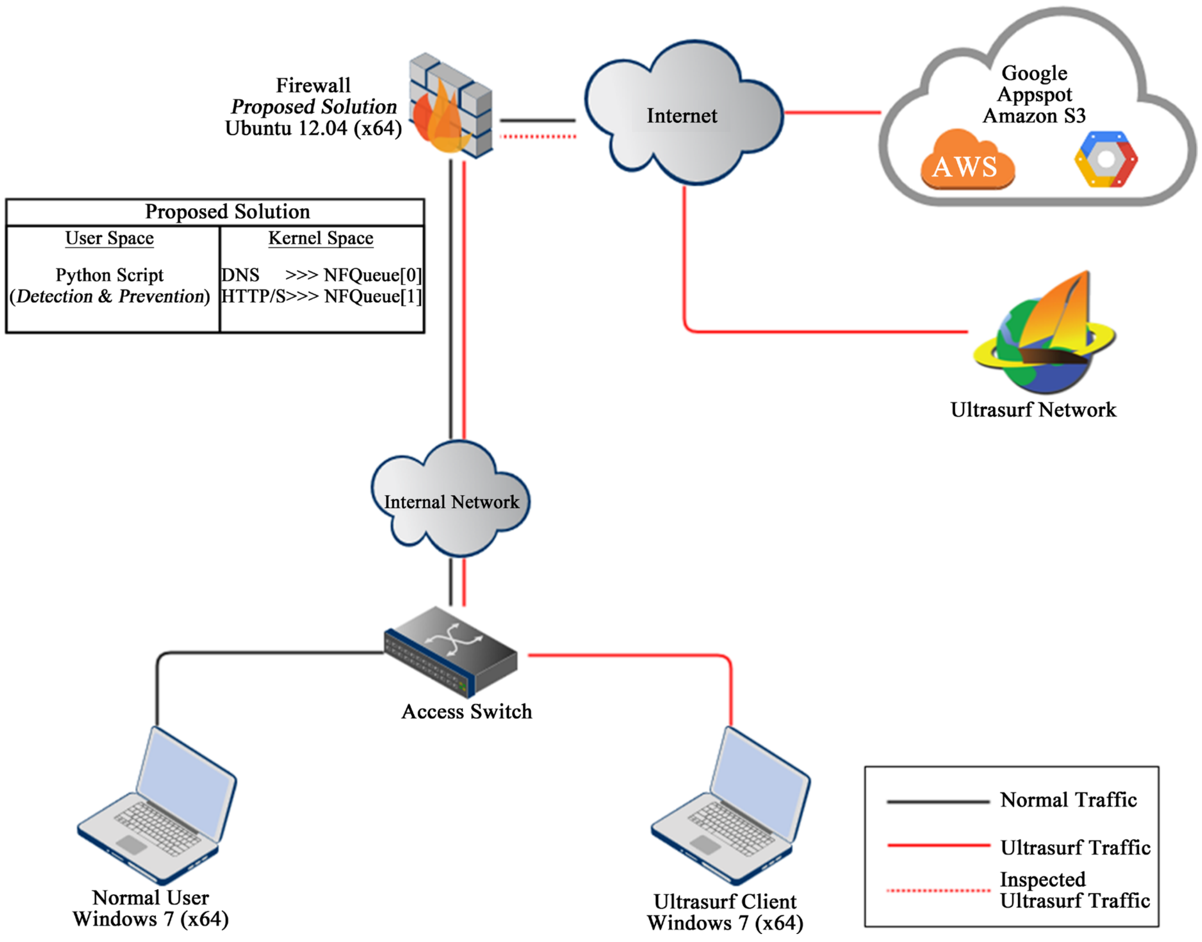


Figure 3. Architecture of the implementation environment.

Table 1. Overall successful detection ratio over delay period of n seconds.

n = 1 s	*n* = 2 s	*n* = 3 s	*n* = 4 s	*n* = 5 s	*n* = 6 s	*n* = 7 s	*n* = 8 s	*n* = 9 s	*n* = 10 s
32.12%	39.09%	**71.82%**	72.73%	73.64%	73.68%	74.85%	88.79%	95.76%	96.36%

As discussed previously, Ultrasurf client is capable of detecting and bypassing DNS censorship. The proposed design exploits this feature and deploys a system that mimics DNS censorship, thus, forcing Ultrasurf client to try to bypass this fake censorship by initiating an encrypted TCP connection on port 443 with one of its hard-coded proxy servers; thus, using this behavior to detect and block Ultrasurf traffic. The proposed solution, is capable of detecting this behavior and respond by rerouting the traffic from the host running Ultrasurf client to a black hole.

Unlike [25], the proposed solution dose not rely on blocking all direct TCP connections on port 80 and 443 (*i.e.*, by IP address), rather than detecting Ultrasurf traffic based on its client application behavior. In this research, we focused on studying the behavior of Ultrasurf client version 13.04, which at that time, was the most recent version. Notably, in later versions of Ultrasurf client the behavior described in this article behavior is no longer consistent.

5. Conclusion

In this study, the issue of using anti-censorship applications in small and medium size enterprises was addressed. We focused on Ultrasurf; a program used encryption in single hop proxy architecture to bypass censorship. We analyzed its behavior, classified its traffic and proposed a behavioral-based solution to detect and block the Ultrasurf traffic at the network level. A stateless packet inspection system was implemented as a PoC for the proposed solution using Python, Linux IP-Tables and NFQueue. The evaluation results showed that the proposed solution was capable of achieving a successful detection and prevention ratio of 71.82% with 3 seconds delay period for DNS queries requesting Google, Appspot, and Amazon.

References

[1] (2015) Tor Project: Anonymity Online. [Online]. https://www.torproject.org/

[2] (2015) Ultrasurf—Free Proxy-Based Internet Privacy and Security Tools. [Online]. http://ultrasurf.us/index.html

[3] (2015) GPass. Softonic. [Online]. http://gpass.en.softonic.com/

[4] (2015) Garden Networks for Information Freedom. [Online]. http://gardennetworks.org/products

[5] (2014) Fire Phoenix Secure Browser. Vionika.

[6] (2007) Global Internet Freedom Consortium. Defeat Internet Censorship: Overview of Advanced Technologies and Products.

[7] Leberknight, C.S., Chiang, M. and Wong, F.M.F. (2012) A Taxonomy of Censors and Anti-Censors Part II: Anti-Censorship Technologies. *International Journal of E-Politics*, **3**, 20-35. http://dx.doi.org/10.4018/jep.2012100102

[8] Appelbaum, J. (2012) Technical Analysis of the Ultrasurf Proxying Software. The Tor Project, Technical Report.

[9] (2015) The netfilter.org "iptables" Project. [Online]. http://www.netfilter.org/projects/iptables/

[10] Rovniagin, D. and Wool, A. (2011) The Geometric Efficient Matching Algorithm for Firewalls. *IEEE Transactions on Dependable and Secure Computing*, **8**, 147-159. http://dx.doi.org/10.1109/TDSC.2009.28

[11] Houmansadr, A., Nguyen, G.T., Caesar, M. and Borisov, N. (2011) Cirripede: Circumvention Infrastructure Using Router Redirection with Plausible Deniability. *Proceedings of the* 18*th ACM Conference on Computer and Communications Security*, Chicago, 17-21 October 2011, 187-200. http://dx.doi.org/10.1145/2046707.2046730

[12] Wang, Q., Gong, X., Nguyen, G.T., Houmansadr, A. and Borisov, N. (2012) Censorspoofer: Asymmetric Communication Using Ip Spoofing for Censorship-Resistant Web Browsing. *Proceedings of the* 2012 *ACM Conference on Computer and Communications Security*, New York, 16-18 October 2012, 121-132. http://dx.doi.org/10.1145/2382196.2382212

[13] Houmansadr, A., Brubaker, C. and Shmatikov, V. (2013) The Parrot Is Dead: Observing Unobservable Network Communications. *Proceedings of the* 2013 *IEEE Symposium on Security and Privacy*, Berkeley, 19-22 May 2013, 65-79. http://dx.doi.org/10.1109/sp.2013.14

[14] Geddes, J., Schuchard, M. and Hopper, N. (2013) Cover Your ACKs: Pitfalls of Covert Channel Censorship Circum-

vention. *Proceedings of the* 2013 *ACM SIGSAC Conference on Computer & Communications Security*, Berlin, 4-8 November 2013, 361-372.

[15] Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Cheung, S., Wang, F. and Boneh, D. (2012) StegoTorus: A Camouflage Proxy for the Tor Anonymity System. *Proceedings of the* 2012 *ACM Conference on Computer and Communications Security*, New York, 16-18 October 2012, 109-120. http://dx.doi.org/10.1145/2382196.2382211

[16] Burnett, S. and Feamster, N. (2013) Making Sense of Internet Censorship: A New Frontier for Internet Measurement. *ACM SIGCOMM Computer Communication Review*, **43**, 84-89. http://dx.doi.org/10.1145/2500098.2500111

[17] Jones, B., Lee, T.-W., Feamster, N. and Gill, P. (2014) Automated Detection and Fingerprinting of Censorship Block Pages. *Proceedings of the* 2014 *ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, 3-7 November 2014, 299-304.

[18] Anderson, D. (2012) Splinternet behind the Great Firewall of China. *Queue*, **10**, 40.

[19] Khattak, S., Javed, M., Khayam, S.A., Uzmi, Z.A. and Paxson, V. (2014) A Look at the Consequences of Internet Censorship Through an ISP Lens. *Proceedings of the* 2014 *Conference on Internet Measurement Conference*, Vancouver, 5-7 November 2014, 271-284.

[20] (2015) OpSyria: When the Internet Does Not Let Citizens down. Reflets. http://reflets.info/opsyria-when-the-internet-does-not-let-citizens-down/

[21] Chaabane, A., Chen, T., Cunche, M., De Cristofaro, E., Friedman, A. and Kaafar, M.A. (2014) Censorship in the Wild: Analyzing Internet Filtering in Syria. *Proceedings of the* 2014 *Conference on Internet Measurement Conference*, Vancouver, 5-7 November 2014, 285-298.

[22] (2010) Brita, Digital Weapons Help Dissidents Punch Holes in China's Great Firewall, WIRED. http://www.wired.com/2010/11/ff_firewallfighters/

[23] Callanan, C., Dries-Ziekenheiner, H., Escudero-Pascual, A., and Guerra, R. (2014) Leaping over the Firewall: A Review of Censorship Circumvention Tools.

[24] (2012) Tor's Critique of Ultrasurf: A Reply from the Ultrasurf Developers. Ultrareach Internet Corp.

[25] Osman, B., Abas, A. and Harmoni, K. (2011) Strategy to Block Traffic Create By Anti-Censorship Software in LAN for Small and Medium Organisation. *Proceedings of the* 3rd *International Conference on Computing and Informatics* (*ICOCI*), Bandung, 8-9 June 2011, 358-365.

[26] (2015) Squid: Optimising Web Delivery. http://www.squid-cache.org/

[27] (2015) The Netfilter.org "Libnetfilter_Queue" Project. http://www.netfilter.org/projects/libnetfilter_queue/