# MANTICORE: Masking All Network Traffic via IP Concealment with OpenVPN Relaying to EC2

Patrick Butler, Adam Rhodes, and Ragib Hasan
*Department of Computer and Information Sciences*
*University of Alabama at Birmingham*
*Birmingham, AL 35294, USA*
{*pbutler, adrhodes, ragib*}*@uab.edu*

***Abstract*—Malware and computer forensic researchers often communicate with malicious servers, either directly or indirectly, through the web browser or other ports utilized by malicious software. Communication with this form of adversary can sometimes necessitate the use of a proxy server in order to conceal the true origin of the researcher's traffic. Open source projects such as OpenVPN currently offer a structured method for establishing software based virtual private networks (VPNs) between arbitrary clients and servers. Likewise, paradigms exist which allow a user to proxy traffic from one end of a VPN to another, effectively masking the origin of traffic being sent to and from the client system. In this paper, we present MANTICORE – a system that combines ideas from VPN with the instancing functionality of a cloud computing system in order to dynamically mask and reassign the apparent IP address of a researcher's system. We also present experimental evaluation of our system on Amazon's Elastic Compute Cloud (EC2).**

***Keywords*-cloud computing; security; forensics;**

## I. Introduction

There are currently a myriad of available servers online which are intended to conceal the IP address of a user's machine. Web-based URL redirectors can provide proxy functionality, but have fixed IP addresses that can be blacklisted by malicious servers such as phishing and malware websites. Similarly, some options such as Tor servers, have well known IPs, which can be easily blocked by malicious servers. In fact, many phishing websites contain code that monitors repeat visitors and blocks the IP addresses used by suspected anti-phishing probes. To be able to thwart such evasive maneuvers, it is necessary to find an efficient, low-overhead, and low cost IP concealment service.

In this paper, we propose using both dynamic provisioning of cloud instances and a virtual private network (VPN) to provide an IP address concealment service for malware researchers. In order to ensure that the proxy server(s) used by the client are both safe and reliable, we utilize machines connected to a cloud infrastructure.

The main contribution of this paper is to demonstrate that such a concept is indeed feasible using commercial clouds, and that the technique achieves comparable or better performance than traditional approaches. By making use of a cloud infrastructure, we have access to a large number of independent IP addresses. Using our system, a researcher is able to dynamically reassign the proxy server in use, which is selected from a pool of available addresses on the cloud. Because of the large scale of most cloud infrastructures, this IP could have a vastly different geographical location than previous proxies used by the client. In practice, this means that if an administrator of a malicious server notices a researcher's interaction with their system and they block traffic from the originating IP address, the researcher may dynamically switch his IP to another address in the cloud infrastructure, and therefore bypass the block.

The rest of the paper is organized as follows: Section II provides background information and overview of related research. Section III provides an overview of our system architecture. Experimental evaluation of our proof-of-concept system is presented in Section IV, and the findings are discussed in Section V. Finally, we discuss future directions in Section VI and conclude in Section VII.

## II. Background and Related Work

Many researchers have used different techniques to accomplish the same basic goal as the one intended for our system. Anonymity and anonymous routing on the Internet are not new ideas. One of the most popular and effective techniques of anonymous routing is onion routing. Originally developed by the Naval Research Laboratory, onion routing involves many encrypted hops between the real source and destination [1], [2]. Its suitability for anonymous connections is a main motivation for the research that has been put into it [3]. Onion routing is well known, but is still being improved upon even today [4].

One of the most well known onion routing services is Tor [5]. Tor has many useful features and is actively being developed [6]. Its success is also a problem. Because of its high profile nature, many attacks have been tested against it [7]–[9]. Another large problem with Tor for our purposes is that many malicious servers routinely blacklist the well known Tor server IPs. If a researcher cannot connect to a malicious server due to blacklisting of a Tor server, then anonymity is useless.

978-0-7695-4755-8/12 $26.00 © 2012 IEEE
DOI 10.1109/CLOUD.2012.29

Two other well known proxy methods are web based proxies and browser based proxies. Browser based proxies are straightforward. In a browser's connection settings the IP of a proxy is stored. This method forwards all the traffic from the browser through the proxy. However, it does not route other traffic, such as SSH or FTP through the proxy.

Web based proxies function in a slightly different manner. When a client goes to a website that acts as a web proxy, she is presented with a url prompt. The webpage the client requests is then retrieved by the proxy. After adding the webpage to an iframe and modifying the links, the page is presented to the user. This is even more restrictive than a browser based proxy. Only webpages requested by the client through the web proxy are actually routed through the web proxy. When a researcher wants to connect to a malware command and control server, often browser traffic is not what is used. SSH is just one example of the many forms of communication that might be used. Also, the communication is often on different ports than browser traffic. This necessitates all traffic on all ports being routed through the proxy in order for a system to be useful in this application.

The final type of proxy system is a VPN based proxy. The VPN is set up with a client server model. The server is set up to wrap and forward all traffic from the client and pass incoming trafffic back to the client. This has the benefit of forwarding all traffic on all ports. It also has the benefit of creating a secure connection between the client and the server. This prevents a malicious malware server from performing a man-in-the-middle attack against the researcher. The largest problem with a traditional VPN based proxy is the static nature of the server. As discussed, if the malicious server recognizes the same IP doing many things, it is highly likely to be blocked. A traditional VPN based proxy has no provision for changing its forward facing IP. It does not matter that the malicious server does not know the researcher's IP if the researcher's proxy has been blocked.

Our system is an extension of a VPN based proxy that leverages the elastic nature of the cloud. We utilize two main systems: Amazon's Elastic Compute Cloud (EC2) [10] and OpenVPN [11], [12]. Amazon's EC2 is a very popular cloud computing service provider. It allows users to instantiate virtual machines (VMs) on demand. It is this virtualization that makes EC2, or another cloud provider, advantageous. Each time a virtual machine is instantiated, it is assigned a unique IP based on Amazon's proprietary algorithm. For our purposes, they are psuedo-random, with a low probability of getting the same IP by restarting a VM, despite Amazon only having certain IP ranges available. This allows a user to gain access to many IPs for a very low cost.

OpenVPN is an award winning open source VPN solution [11], [12]. With servers and clients that run on Windows, Mac, and Linux, it is an extremely popular tool. It can create either a layer-3 based IP tunnel (TUN), or a layer-2 based Ethernet TAP that can carry any type of Ethernet traffic running over User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) transports. OpenVPN runs a custom security protocol based on SSL and TLS [13]. It also allows for forwarding on both the client side and server side. IPsec is the other major VPN protocol [14], [15]. It is widely used by many vendors, but is much more complex so we have favored OpenVPN. Many others have used some of the techniques we are using. Because of a VPN's strong security, it is desirable to have an easy way to connect two separate locations securely. Most research has been on making the process easier while also making it more dynamic, flexible, and programmable. Some of this work can be found in [16]–[18]. Liu et al. propose using a VPN with a cloud in a slightly different manner [19]. While we propose to use the elastic nature of the cloud to dynamically change the external IP of a VPN server, they propose to use the elastic nature to serve clients with a VPN Software-as-a-service (SaaS) with a pay as you go model.

As mentioned previously, our method differs substantially from these approaches. Using the dynamic nature of the cloud to easily change our perceived IP and OpenVPN to secure the connection from the client to the server, we can hide our identity while investigating malicious servers. These two technologies have yet to be combined in this way.

## III. System Design

In this section, we discuss details of our system model and its different components.

### A. Threat Model

The system assumes that the cloud service provider is a trustworthy entity. It also assumes that any adversary involved is a passive one that is unable to eavesdrop between the client and the proxy server because of the assumed security provided by the established VPN between them. The main adversary is the malicious site or server that the researcher trying to explore. The adversary can monitor and record the IP addresses of incoming connections. The adversary's goal is to identify possible anti-phishing or anti-malware researchers and blacklist their IP addresses.

### B. System Architecture

We designed our system using several widely used off-the-shelf tools and open source components. The servers were set up on Amazon EC2's micro instances, which are available on the free tier. Each one was set up with an OpenVPN server that ran on startup. The server uses a normal server configuration except it pushes a free public DNS to the client. The network address translation (NAT) table in the firewall was modified to forward packets with masquerading between the outside connection and the VPN connection.

On the client machine, an OpenVPN client was set up to communicate with the servers. The one option of note is the "redirect-gateway". This sets up the forwarding of

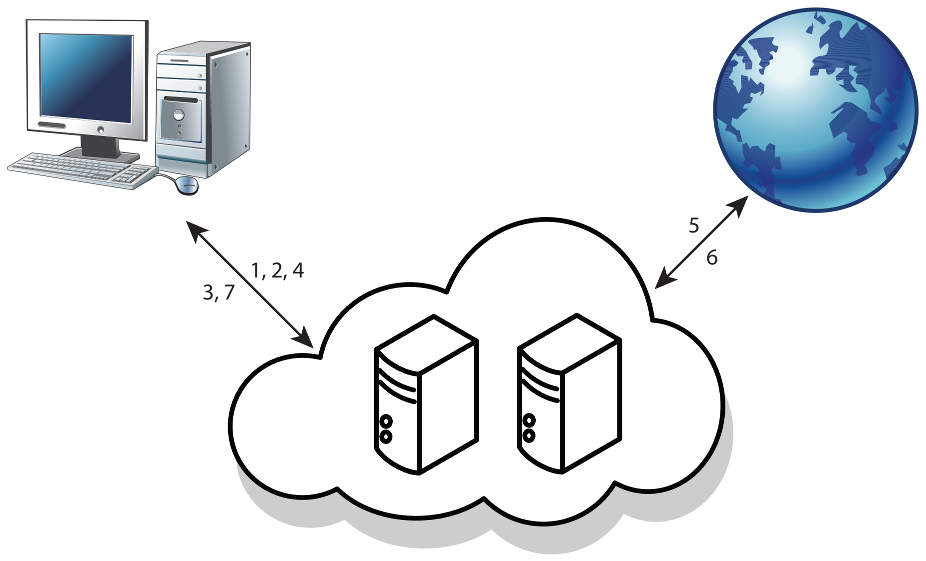

Figure 1. **Our system setup, startup, and basic usage. 1.) The researcher starts the instances on EC2. 2.) The researcher initiates a VPN connection with a randomly chosen instance. 3.) The OpenVPN server on the instance responds to complete the VPN connection. 4.) The researcher requests data from a server on the internet, such as a malicious one, via the VPN. 5.) The instance wraps the request and sends it on to the actual server. 6.) The server responds to the instance. 7.) The instance sends the response on to the researcher.**

traffic through the VPN on the client side. Otherwise, it was a standard OpenVPN client configuration.

Also on the client is the main program. Using Amazon's Java AWS SDK, it automates the entire process. After pointing to all the required certificates and keys, the user can press start. This starts up the VMs, waits for the bootup process to finish, and then starts OpenVPN with the parameters to connect to the OpenVPN server of a random instance. Then a user can at any time switch or stop. Switching disconnects from the VPN, picks another random server, and connects to its VPN server. Then it restarts the instance that is no longer being used. Stopping disconnects from the VPN and stops all instances. The program also shows the current IP that the user is masquerading as. The basic steps in starting MANTICORE and sending data are diagramed in Figure 1.

## IV. Evaluation

To test the feasibility of using a cloud based VPN, we developed a proof of concept implementation. For our tests, we used two cloud instances. This allows for fast switching and cuts down on the potential costs of running the instances. All of the testing of the system was done on the free tier of Amazon Web Service (AWS) at no cost. The client was a HP Envy 14 running Windows 7. All of the tests were done with two micro instances running at Amazon's Virginia data center.

### A. *Operation Times*

The first set of tests are designed to measure how much time each of the three basic operations took under varying conditions. Thirty different runs each consist of the startup operation, one switch operation, and finally a stop operation. The time for each operation in each run was recorded. These tests gave us an idea on not only how long MANTICORE took to start and stop but also approximately how long it would take to get a new IP.

Once our operation time tests were completed, we first graphed the individual runs, as seen in Figure 2. There was a high amount of variability between the runs. No effort was made to eliminate normal outside factors such as other connections on the same WIFI network. Run fourteen in particular has an abnormally high switch time. We are not sure of the cause of this high switch time. While Figure 2 shows a lot of data, Figure 3 shows the average time of each operation. The vertical bars represent one standard deviation from the mean. As we can see, stopping is the fastest operation and also the one with the least variability. Starting is the slowest operation, but the variability is not too high. Switching takes a medium amount of time but has high variability.

### B. *Latency Penalty*

We also wanted to run tests in order to determine how much of a penalty is incurred by using MANTICORE as opposed to a direct connection between the server and user. Four web-servers were selected: Google, China Daily, and two Skype servers. The Google server is located in Seattle,

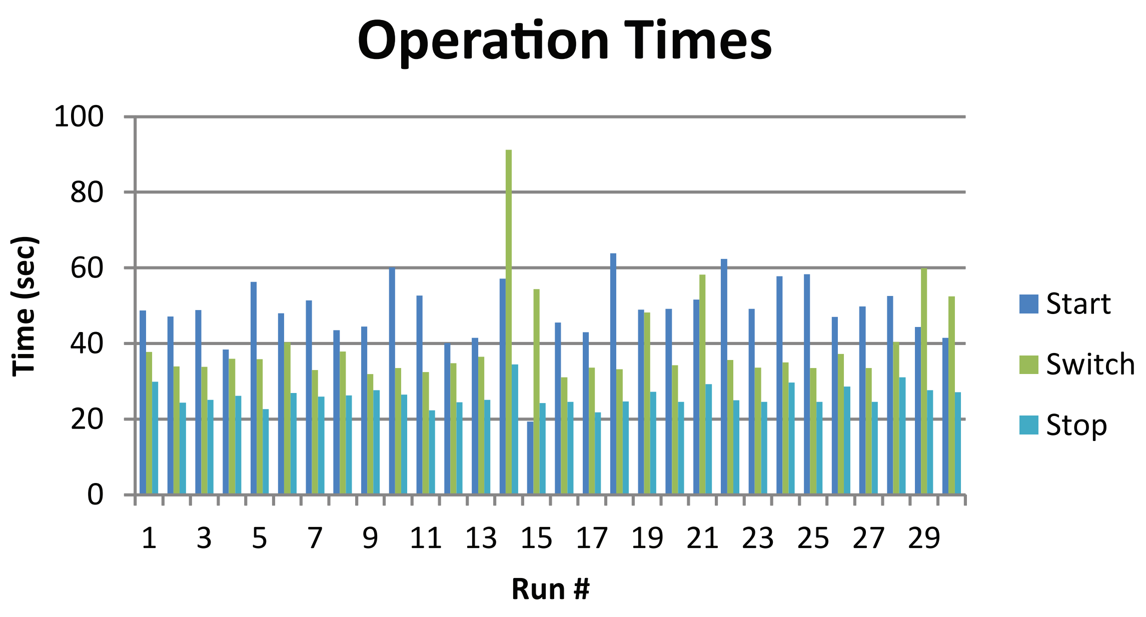

Figure 2. **Start, switch, and stop times for 30 separate runs.**

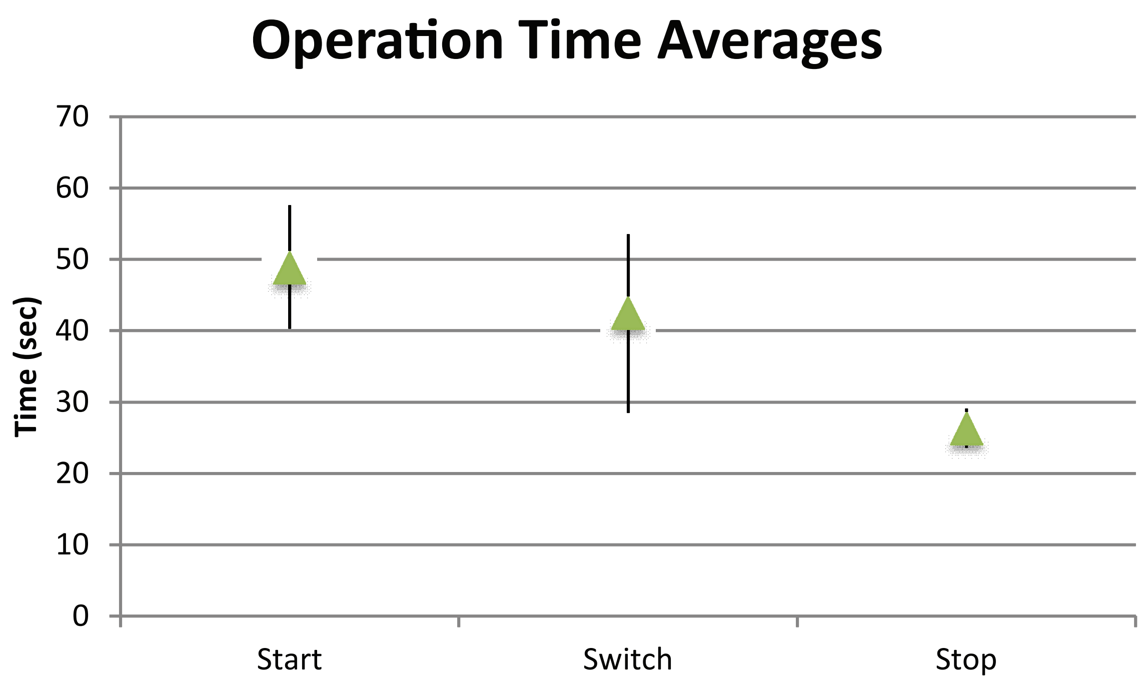

Figure 3. **Average start, switch, and stop times.**

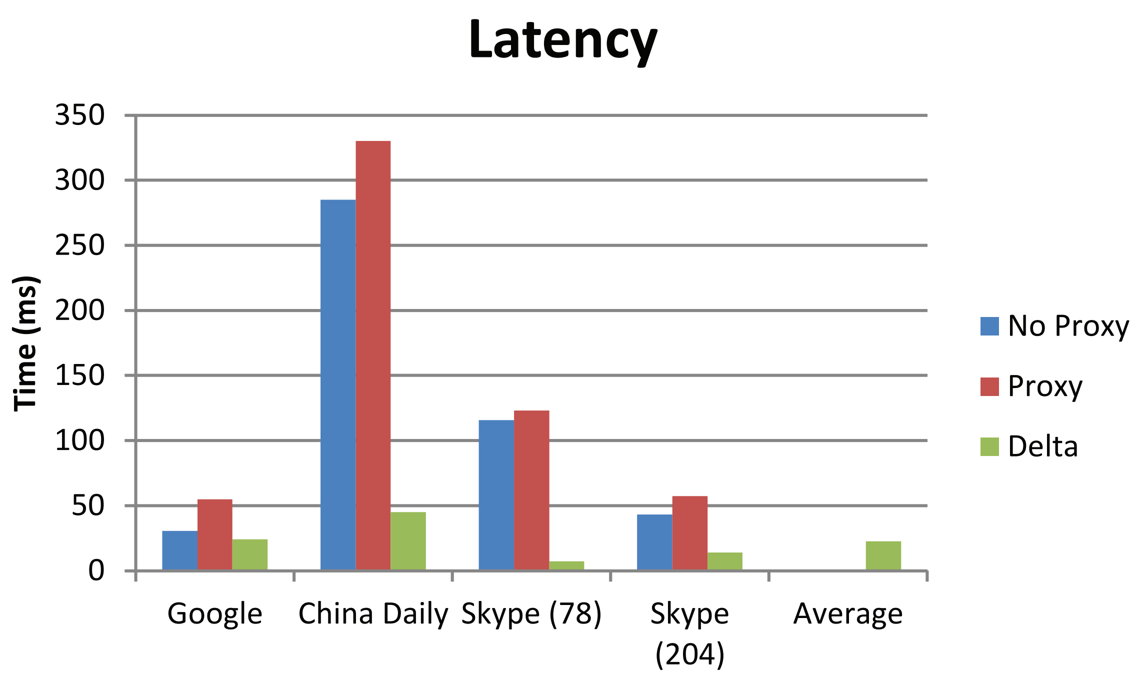

Figure 4. **Average latencies over 1000 pings to each server normally and through MANTICORE. "Delta" reflects the difference between the two latencies.**

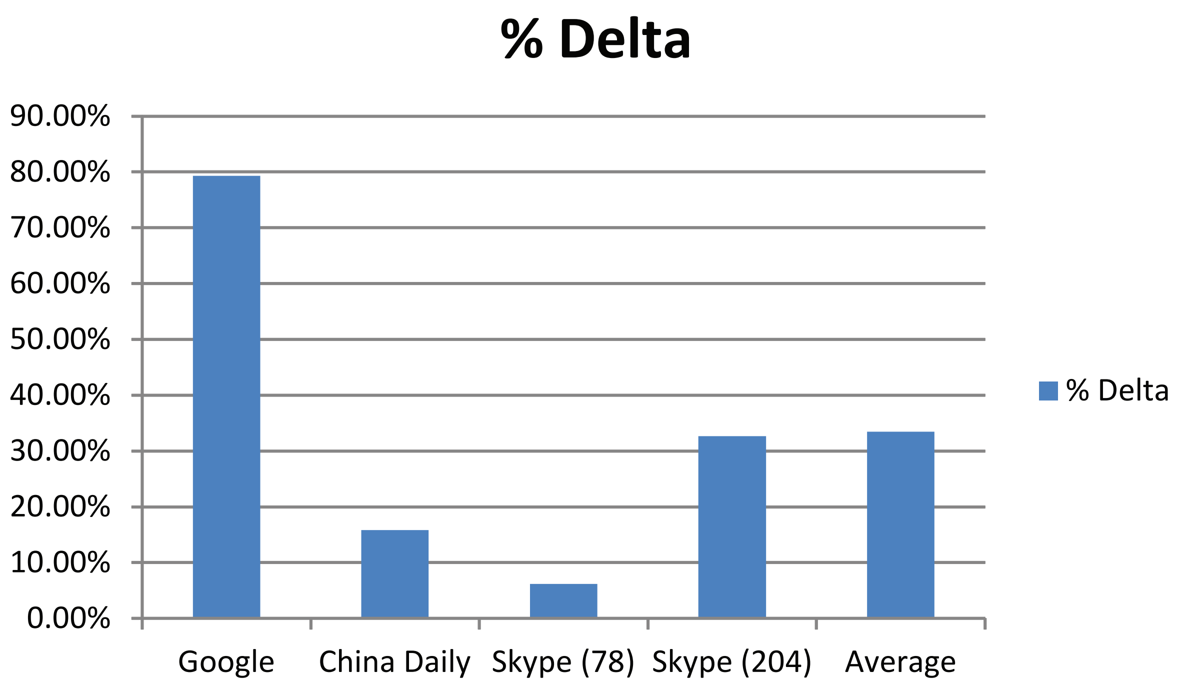

Figure 5. **Average percentage increase in latency using MANTICORE compared to a normal setup.**

Washington. The China Daily server is located in Beijing, China. The Skype server with an IP of 204.9.163.162 is located in Toronto, Canada. The Skype server with an IP of 78.141.177.7 is located in Berlin, Germany. Each server was pinged one thousand times normally and one thousand times through MANTICORE.

After running the latency tests, the average latency for each server with MANTICORE and without MANTICORE was calculated. Next, the difference between the latency with MANTICORE and normal was calculated. Finally, the average of the differences was calculated. These results are shown in Figure 5. Here, the average latency penalty is around 22ms. This is very low and does not represent a large portion of the total latency. Figure 4 shows how much of the latency is incurred by MANTICORE. The tests with Google appear to denote a poor performance by percentage, but notice that the latency to Google started around 30ms, so any increase would appear rather large. The average penalty percentage is under 33%.

## V. Discussion

### A. Analysis of Results

The results discussed in the above section can be accounted for when we observe the location of our proxy servers. Since our proxy is located in Virginia, this means that all traffic must initially make several mandatory hops to reach the proxy server before proceeding to the intended destination. In practice, this means that if the geo-location of our proxy is between the client location and the final destination, there should not be a notable decrease in MANTICORE's performance when compared to a client connecting to the intended destination directly using a traditional server-client model. In contrast, when the shortest available path (as determined by the acting intermediary routers) to the proxy server transfers the network traffic away from its intended destination, the proxy server must then send the traffic back towards the final destination, causing it to potentially retrace a portion of the path it took to arrive at the proxy originally.

### B. Cost Analysis

When attempting to estimate the running cost for MANTICORE, it is important to consider that the bandwidth usage for its intended purpose is very small. It is not unreasonable to assume that, if MANTICORE is used only for connecting to malicious command and control servers through the use of malware, less than 2 gigabytes of bandwidth will be used in a single month by MANTICORE. In fact, this bandwidth can be considered a reasonable upper bound even for a large number of samples of malware. As of January 2012, Amazon charges \$0.12 per gigabyte (after the first gigabyte) of bandwidth used after each month. Note that this bandwidth only applies to outbound traffic, inbound traffic is not charged. Amazon also charges \$0.02 per hour of running time for a micro on-demand instance on the EC2 cloud.

In the current model, MANTICORE operates with two instances simultaneously which effectively increases the cost of running the system to \$0.04 an hour. It is currently assumed that an analyst will be present when MANTICORE is in use, so it is reasonable to assume that a single instance of MANTICORE will be operating 40 hours or less each week. Under these assumptions, the operating costs of MANTICORE come to roughly \$6.52 a month.

We now consider a more costly approach to the usage of MANTICORE: the malware analyst instead uses MANTICORE as part of an automated system that operates 24 hours a day. For this approach, we assume that the upper bound for outbound bandwidth from the EC2 server approaches, but does not exceed, 6 gigabytes, resulting in a \$0.60 cost per month at the rate mentioned above. We know that the operating cost for two micro-instances which run 24 hours a day is \$26.88 a month. Adding these costs gives us a total cost of only \$27.48 for operating a single instance of MANTICORE 24 hours a day for a month. This is still a small expense compared to the advantage MANTICORE provides to malware researchers.

## VI. Future Work

In future experiments, we hope to test methods to improve the economic viability of the system as well as to explore more methods for obtaining a larger variety of IP addresses for the proxy server. In particular, we intend to experiment with use-cases that would allow multiple clients to connect to a VM instance on a proxy server in order to share its bandwidth and thus reduce the overall cost of running multiple clients through MANTICORE. To experiment with improving the range of IPs used by the system, we may also run trials in which servers in multiple geo-locations are accessible to MANTICORE through Amazon EC2. Additionally, we desire to develop functionality for MANTICORE which allows the system to be used over multiple cloud service providers.

## VII. Conclusion

In this paper, we have presented MANTICORE – a system for dynamically masking one's IP by utilizing the dynamic and flexible nature of the cloud. MANTICORE is extremely flexible. The instances can be located at any of Amazon's EC2 datacenters. Any number of instances may be used in order to speed up the switching. Due to the low compute and bandwidth requirements of the original problem, there is an extremely low cost to operate the system. The flexible and dynamic nature of the cloud allows for fast IP switching. The latency penalty of running through MANTICORE is very low. The connection to the proxy server is secure, preventing man-in-the-middle snooping and attacks. Finally, this is a legal method which is capable of utilizing a large, mostly unknown set of IPs that is of a size which is likely rivaled by only the largest of botnets.

## Acknowledgment

The authors would like to thank the UAB Digital Forensics Center for allowing access to the facilities and infrastructure at the UAB Phishing Database. Hasan was supported by Amazon through an Amazon research grant, by Google through a Google Faculty Research Award, by the Department of Homeland Security, and by the Office of Naval Research via Grant #N000141210217.

## References

[1] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[2] M. Reed, P. Syverson, and D. Goldschlag, "Proxies for anonymous routing," in *Proc. of ACSAC*, 1996, pp. 95–104.

[3] P. Syverson, D. Goldschlag, and M. Reed, "Anonymous connections and onion routing," in *Proc. of IEEE S & P*, 1997, pp. 44–54.

[4] S. Katti, D. Katabi, and K. Puchala, "Slicing the onion: Anonymous routing without PKI," in *Proc. of ACM HotNets*, 2005.

[5] J. Clark, P. Van Oorschot, and C. Adams, "Usability of anonymous web browsing: an examination of Tor interfaces and deployability," in *Proc. of SOUPS*. ACM, 2007, pp. 41–51.

[6] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," in *Proc. of ACM CCS*, 2010, pp. 329–339.

[7] T. Abbott, K. Lai, M. Lieberman, and E. Price, "Browser-based attacks on Tor," in *Proc. of PET*, 2007, pp. 184–199.

[8] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against tor," in *Proc of WPES*. ACM, 2007, pp. 11–20.

[9] X. Wang, J. Luo, M. Yang, and Z. Ling, "A potential HTTP-based application-level attack against Tor," *Future Generation Computer Systems*, vol. 27, no. 1, pp. 67–77, 2011.

[10] "Amazon Elastic Compute Cloud (EC2)," In http://aws.amazon.com/ec2.

[11] P. Ferguson and G. Huston, "What is a VPN?" in *Proc. of OPENSIG*, 1998.

[12] "Facts about OpenVPN," In http://openvpn.net/index.php/about/openvpn-facts.htm.

[13] "OpenVPN security overview," In http://openvpn.net/index.php/open-source/documentation/security-overview.htm.

[14] "IPSec charter," In http://datatracker.ietf.org/wg/ipsec/charter/.

[15] J. Ioannides and A. Keromytis, "Network security and IPsec (tutorial)," in *Proc. of ACM CCS*, 2000, p. 11.

[16] H. Hiroaki, Y. Kamizuru, A. Honda, T. Hashimoto, K. Shimizu, and H. Yao, "Dynamic IP-VPN architecture for cloud computing," in *Proc. of IEEE APSITT*. IEEE, 2010, pp. 1–5.

[17] R. Isaacs, "Lightweight, dynamic and programmable virtual private networks," in *Proc. of IEEE OPENARCH*, 2000, pp. 3–12.

[18] P. Lago and R. Scandariato, "A TINA-based solution for dynamic VPN provisioning on heterogeneous networks," in *Proc. of IEEE TINA*, 2000, pp. 13–15.

[19] Q. Liu and W. Gu, "An elastic public vpn service model based on cloud computing," in *Proc. of IEEE ICSESS*, 2011, pp. 290–294.