Attacking Tor through Unpopular Ports

Muhammad Aliyu Sulaiman
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
muhalisu@kfupm.edu.sa

Sami Zhioua
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
zhioua@kfupm.edu.sa

***Abstract*—Anonymity systems try to conceal the relationship between the communicating entities in network communication. Popular systems, such as Tor and JAP, achieve anonymity by forwarding the traffic through a sequence of relays. In particular, Tor protocol constructs a circuit of typically 3 relays such as no single relay knows both the source and destination of the traffic. A known attack on Tor consists in injecting a set of compromised relays and wait until a Tor client picks two of them as entry (first) and exit (last) relays. With the currently large number of relays, this attack is not scalable anymore. In this paper, we take advantage of the presence of unpopular ports in Tor network to significantly increase the scalability of the attack: instead of injecting typical Tor relays (with the default exit policy), we inject only relays allowing unpopular ports. Since only a small fraction of Tor relays allow unpopular ports, the compromised relays will outnumber the valid ones. We show how Tor traffic can be redirected through unpopular ports. The experimental analysis shows that by injecting a relatively small number of compromised relays (30 pairs of relays) allowing a certain unpopular port, more than 50% of constructed circuits are compromised.**

***Keywords*-Anonymity Systems; Tor Network; Privacy; Network Security;**

I. Introduction

Anonymity systems, such as Tor [1] and Jap [2] are designed primarily to provide privacy and anonymity to censored Internet users. These systems achieve anonymity by embedding user data inside several layers of encryption and by forwarding the traffic through a set of relay nodes/proxies. This makes the job of an eavesdropping adversary much more challenging since by just observing the traffic she cannot deduce who is communicating with whom and what is the type of traffic exchanged.

Tor [1], [3] represents the current state-of-the-art in low-latency anonymity systems. The Tor network is currently the largest deployed anonymity network ever, consisting of almost 3000 relays and more than an estimated 300,000 users.

Tor provides anonymity to its users by forwarding the traffic through a sequence of relays (Tor node and Tor relay are used interchangeably in this paper). No one of these relays is able to link the source and the destination of the traffic: the first (entry) relay knows the source but not the destination, the last (exit) relay knows the destination but not the source and any intermediate relay knows neither the source nor the destination. For an attacker to break the anonymity of Tor, it is enough to observe the traffic of the entry and exit relays. Indeed, if the entry and exit relays traffic is observed, a simple traffic analysis can easily confirm who is communicating with whom. Since relays forwarding the Tor traffic are typical user machines run by volunteers, a popular attack in the literature is to inject a certain number of malicious relays in the Tor network and then wait for the Tor client to pick two of them as entry and exit relays. Several variants of this attack exist in the literature. The main limitation of this attack is its scalability: with a couple of hundreds of relays the attack is efficient however, with the 3000 relays of the current Tor network, the attack is not scalable anymore.

In this paper, we revisit this same attack but improving significantly its scalability. The main idea is to take advantage of unpopular ports in Tor to maximize the chances of the success of the attack. Unpopular ports are port numbers assigned to applications that are not recommended with Tor. For instance, Tor blocks by default all peer-to-peer applications related port numbers (BitTorrent, Kazaa, etc.) because of their bandwidth greediness [4], [5], [6]. Hence, Tor default exit policy rejects relay requests if they try to use one of those unpopular ports. However, since Tor is volunteer-based anonymity protocol in which individual users donate bandwidth to relay traffic, it is completely the responsibility of relay donors to decide how they contribute to the Tor network; some relay donors decide to accept such unpopular ports in their exit policies. This makes the number of relays allowing unpopular ports relatively small and exposes the anonymity of Tor system. In this paper we propose an attack that takes advantage of the small number of relays allowing unpopular ports to significantly increase the scalability of the above attacks.

Unpopular ports in Tor have been used in a previous work [7] to compromise the anonymity of Tor. The main difference, however, is that in this paper we actively take advantage of the presence of unpopular ports by pushing

The authors would like to acknowledge the support provided by the Deanship of Scientific Research at King Fahd University of Petroleum and Minerals (KFUPM) under research grant FT111003

a Tor client victim to open connections through unpopular ports. Previous work consisted simply to passively assess the impact of using unpopular applications on the anonymity of Tor.

The rest of the paper is organized as follows; Section 2 surveys related work. Section 3 gives an overview of Tor protocol. Section 4 describes in detail the proposed attack, in particular, the attack steps, how to actively redirect connections through unpopular ports, and an empirical analysis on the efficiency of the attack with respect to the number of injected relays. Finally, Section 5 concludes.

II. Related Work

Murdoch and Watson [8] conducted an empirical analysis that investigated the relationship between the path selection algorithm and path compromise with respect to the attacks cost for the adversary. They identified the fraction of malicious Tor routers and the fraction of adversary-controlled bandwidth as important factors for predicting the adversarys ability to compromise paths. Browser based attacks on Tor was presented by Abbott et al. [9], the main idea is that the attacker can trick a users web browser into sending a peculiar signal over the Tor network and subsequently detected using traffic analysis. In the paper, they described how a malicious node acting as exit node, when selected by a client can insert a JavaScript code into unencrypted payload and transmit to the client, which then run on client machine and can generate identifiable signal pattern that can be detected by the server. Evans et al presented an attack based on legitimate guard node nd malicious exit node [10]. In their attack guard node used by a legitimate client is kept busy through the long path generation. This push client to use malicious exit router to relays its traffic Bauer et al. [11] demonstrated how extend is the routing selection optimization for performance exposed Tor protocol to end-to-end traffic analysis attack from non-global adversaries with minimum resources. Their approach involves compromising guard and exit nodes, by injecting few nodes with high bandwidth and high uptime claims, in a similar work, Bauer et al exploit the role of ports in compromising routing path based on the type of traffic being propagated [7]. In this work we presented a new attack scenario that focus on Tor unpopular ports and combined inspiration from both Abbott et al. [9] and Bauer et al. [11]. Our work is similar to [11], in the sense that we both investigated the relationship of application layer protocol to the resources needed by adversary to compromise path, but differ in the definition of attack scenario; their attack scenario is passive one that involved injecting malicious entry and exit routers with high bandwidth claimed, while in our case is more of active attack to reveal the identity of a Tor client connected to a compromised webserver; the attacker takes advantage of unpopular ports to play on the Tor path selection algorithm to select from few set of relays in which the attacker controlled a significant fractions as explained in section 4. Our attack differ from [9]; theirs is a browser based attack, that trick a users web browser to send a peculiar signals over Tor network, but our involve pushing Tor client to open a new connection through Tor network.

III. Unpopular Ports Attack

A popular attack scenario on Tor is for an adversary to inject as many malicious Tor nodes as possible then to wait for a Tor client (victim) to pick one or several of these to construct his circuits. This scenario was feasible when the Tor network was composed of less than 500 relays. However, currently the Tor network contains more than 3000 relays which makes those attacks not scalable anymore. Our proposed attack is inspired by the same idea. The main difference is that, instead of injecting typical Tor nodes (allowing typical port numbers e.g. 80, 53, etc.), we inject only Tor nodes allowing unpopular ports (e.g. 25, 119, etc.). Since only a small fraction of Tor nodes allow unpopular ports, the malicious injected Tor nodes will outnumber the valid ones which significantly increases the probability to compromise circuits created by Tor clients. Given the few number of Tor nodes allowing unpopular ports, this attack is still scalable with the current Tor network (January 2013). The main requirement for this attack to work is a Tor client that uses an unpopular application (port) through the Tor network. Table I lists the unpopular ports in Tor. In the sequel we show how it is possible to actively push a typical Tor client (using typical ports) to open a connection through Tor using one of the unpopular ports.

A. Attack Steps

In this attack scenario we assume that a client uses Tor network to connect to a compromised server as shown in Figure 1. However the compromised server injects a program into a requested page that will “invite” the client to open another connection through one of the Tor unpopular ports, this way the chances that the client will choose one of the adversary injected malicious exit nodes with high self-advertised bandwidth are significant. The attack proceeds as follows:

1) The attacker controls a compromised web server.
2) The attacker injects a number of malicious exit routers that accept a particular unpopular port with high bandwidth claim.
3) The attacker injects a number of malicious entry routers with high bandwidth claim.
4) A client connects to the compromised web server anonymously using Tor network through a typical port (e.g. 80, 53, etc.).
5) The server injects a hidden stored script into the requested web page.
6) The Tor client machine executes the injected script which forces the opening of a new connection through Tor network to a remote server using unpopular port.

Port	Description
25	Simple Mail Transfer Protocol (SMTP)
119	Network News Transfer Protocol (NNTP)
563	Network News Transfer Protocol over TLS/SSL (NNTPS)
1214	Kazaa Peer 2 Peer (P2P)
6346	Gnutella Peer 2 Peer (P2P)
6347	Gnutella Alternate (P2P)
4661	eDonkey Peer 2 Peer (P2P)
6881	BitTorrent Peer 2 Peer (P2P)
6969	BitTorrent tracker (P2P)

Table I
LIST OF UNPOPULAR PORTS IN TOR PROTOCOL

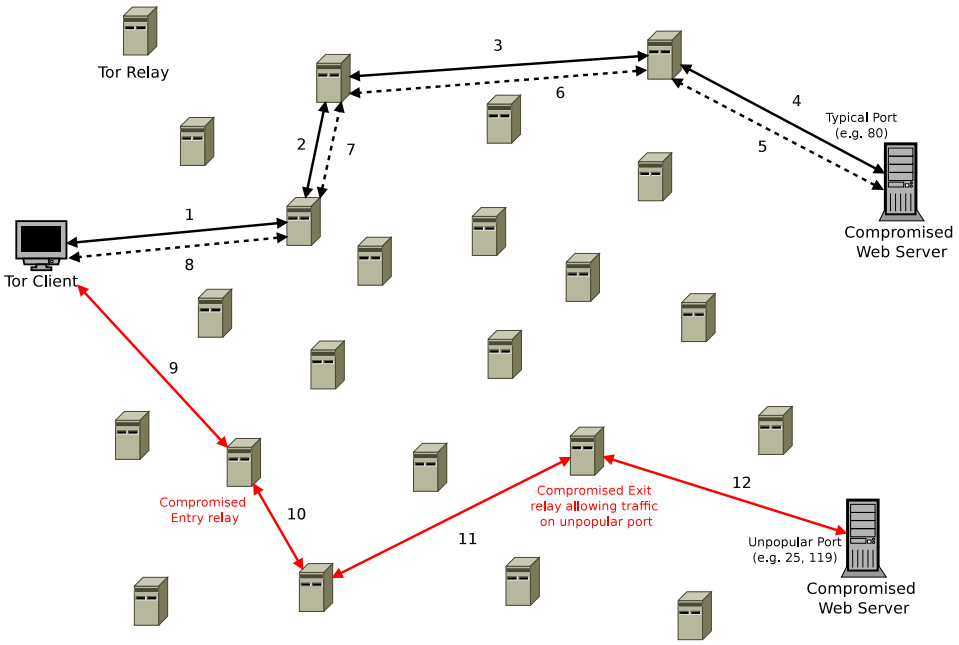

Figure 1. Tor Attack Scenario

7) The attack is successful when the Tor client picks two compromised Tor routers for his circuit entry and exit.

To carry out this attack successfully we need to address some issues such as how could the attacker compromised the webserver? If the attacker succeeded in compromising the webserver and injected the script into the client machine how can the script force the client to open a new connection through Tor network to a remote server using unpopular port?

B. Active Port Redirection

This section discusses how the attacker, using a compromised Web server, can force a Tor client to open a new connection that redirects traffic through one unpopular port. Depending on the type of the attacker, we consider two possible methods allowing to redirect traffic. An external attacker can take advantage of a vulnerability in a web page due to poor programming practices and stores a script into the page. Later, when a Tor client visiting the site receives response that contains a script that will force its browser to open a new connection through an unpopular port. An internal attacker with Webmaster access privilege to the web page can simply include a script into the page.

To demonstrate the effectiveness of this active port redirection, we create two instances of remote virtual machines using Amazon EC2 (Elastic Compute Cloud) cloud services [12]. In the first instance we installed Apache web server to host a simple article web page. In the second instance we installed `socket.io` with `node.js`. `Socket.io` which supports *WebSocket* [13] was the best option for our technique since it has the ability to sense if the users web browser is configured to use a proxy to connect to a remote server and eventually setup a persistent tunnel. Besides, `Socket.io` provides us with a simple way to test redirection for all Tor unpopular ports by changing the listening port of the server. In this regard, we implement a

simple article Web page on the first remote virtual machine. The page consists of the body of the article and text area for visitors to leave comment and a submit button to submit the comments to the back-end database. Within the page we store a JavaScript that push visitors web browser to open a new tab and establish connections with a server-side script based on `socket.io` on `node.js` server which is on the second remote virtual machine. Based on the above setup we launch Tor as client and visit the article web page. Each time we visit the page a new tab is opened and we observe if there is connection to the server-side script on `node.js`.

The experiment was carried out for each one of the unpopular ports. Indeed, for each port, we setup the server-side script to listen to that port and required the injected client-side script to initiate a connection with the remote server. The attack was successful with all Tor unpopular ports except ports 25, 119, and 563. The reason is that these ports are not commonly used for web browsing[1] therefore the Tor enabled web browser intercepted and cancelled the connections.

C. Tor Path Selection Simulation

In order to investigate the relationship of Tor path selection algorithm and the amount of resources needed to make this attack successful, we developed a simulation that adhered to default Tor path selection specification, as provided by the Tor Project. There are two parts of Tor path selection algorithm as elaborated in [14]. The Entrance Router Selection Algorithm which was incorporated into path selection algorithm with the introduction of Entry Guard, in which a client automatically chooses a set of onion routers flagged as fast and stable by the directory servers. The second part of Tor path selection algorithm is the Non-Entrance Router Selection Algorithm, used for selecting subsequent routers in the circuit. In this work we simulate the Non-entrance Path Selection Algorithm since it is optimized to favor router selection with high bandwidth and high uptime.

Non-Entrance Router Selection Algorithm: For network performance reasons, this algorithm is optimized to favor routers with high bandwidth. The algorithm gets as input the routers list and produces a randomly chosen router weighted towards the router advertising the highest bandwidth. At the beginning, the algorithm computes total bandwidth (B) for all the available routers from the router list, subsequently it chooses a pseudo random number C between 1 and B. Each onion router from the list is processed by adding its bandwidth added to a variable T. If the variable T is greater than C, the onion router is selected for inclusion into the path, provided the Tor path selection constraints are met. Otherwise (T is less than C) more onion routers are added to T until T becomes greater than C. From this we can infer that the more bandwidth an onion router self-advertises, the greater the probability that the router is to be chosen. Indeed, the algorithm assigns weights to onion routers based on the probability distribution which is tilted towards the magnitude of the routers self-advertised bandwidth.

The full details of path selection algorithms can be found in [14]. However, router selection algorithm chooses router with the following constraints:

1) All routers in a path must be unique; no router is selected twice for the same path.
2) All routers in a path are chosen from different family, no any router is of the same family with another router in the same path.
3) By default, only one router is chosen from a given /16 subnet.
4) All routers chosen for a path must be running and valid.
5) The first router on the circuit must be flagged as entry guard by a directory server.
6) The exit router selected must support connection to the clients chosen destination host and port.

In all cases, the choice for entry and exit routers are based on considerably large bandwidth; too much bandwidth above one-third of the total bandwidth of all routers in the network may lead to rejection of an onion router, while router with too low bandwidth may not be favored by router selection policy.

D. Injecting Malicious Relays and the Impact on Circuit Construction

Unpopular ports are ports assigned to “not recommended” applications which are typically rejected by Tor relays. These applications are rejected because some of them leak information as they pass through the Tor network due to their unencrypting nature or when doing DNS lookup, and most importantly because some applications require a lot of bandwidth (P2P) and may carry malwares thereby exposing Tor relays to infection. However, since Tor is volunteer-based where individual users donate bandwidth to relay traffic, it is completely the responsibility of relay donors to decide how they intend to make contribution to Tor network. Some relay donors may decide to accept such unpopular ports in their exit policies.

To assess the risk of using unpopular ports in the creation of circuits, we carried out several experiments on all listed unpopular ports as follows. We obtained a snapshot of the active onion routers from Tors directory servers as of January 5th, 2013. We preprocessed the data to obtain information such as each routers name, status, version, self-advertised bandwidth and exit policy and use this data in our simulation. Then using our path selection simulator (Section III-C), we generated for each experiment 1500 circuits and tracked the number of circuits which are compromised.

[1]Port 25 is assigned to SMTP protocol, port 119 is assigned to NNTP (Network News Transfer Protocol), and port 563 is assigned to NNTPS protocol which is NNTP over TLS/SSL.

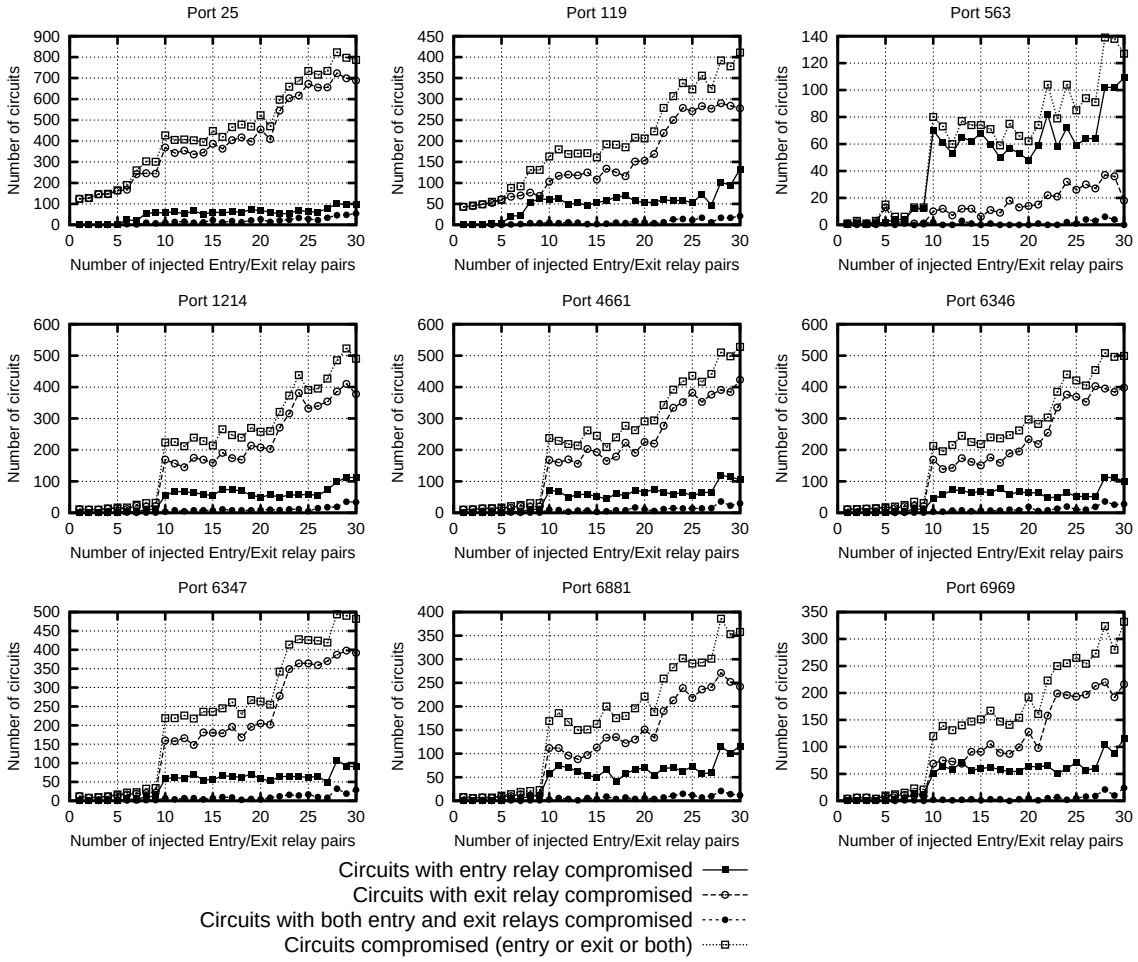

Figure 2. The number of compromised circuits trend while increasing the number of injected relays

In each experiment, we inject a number of malicious routers ranging from 1 pair (entry/exit routers) to 30 pairs. Each malicious router self-advertises 10MB of bandwidth, which is the maximum allowed bandwidth and advertises an exit policy that allows some unpopular ports. Figure 2 shows the result of the experiments where each single experiment is repeated 10 times and the average is plotted. For every experiment (corresponding to a given unpopular port), four values are tracked:

- The number of circuits with entry relay compromised
- The number of circuits with exit relay compromised
- The number of circuits with both entry and exit relays compromised
- The number of circuits compromised (entry relay, exit relay, or both relays compromised.

The port number where our attack has the highest chances of succeeding is 25 (SMTP) since by injecting 30 pairs of compromised routers, 52% of constructed circuits are compromised. The port number with the lowest chances of success of our attack is 563 (NNTPS) where only 9% of the 1500 constructed circuits are compromised. This low success rate can be explained by the fact that several exit relays allow port 563. Hence, injecting malicious relays does not outnumber regular relays allowing to exit on that port.

It can be observed from Figure 2 a certain fluctuation in the path compromise rate as the number of injected malicious routers increases. This is due to random nature

of router selection algorithm which sometimes may not favor routers with higher bandwidth. However overall result shows that path compromise rate increases as the number of injected malicious routers increases in all unpopular ports.

IV. Conclusion

A popular attack on Tor is to inject a certain number of malicious Tor nodes then to wait for a Tor client (victim) to pick one or several of these to construct his circuits. Currently the Tor network contains more than 3000 relays which makes those attacks not scalable anymore. In this paper we described a technique that allows to increase significantly the scalability of this type of attacks. The main idea is: instead of injecting typical Tor nodes (allowing typical port numbers e.g. 80, 53, etc.), we inject only Tor nodes allowing unpopular ports (e.g. 25, 119, etc.). Since only a small fraction of Tor nodes allow unpopular ports, the malicious injected Tor nodes will outnumber the valid ones which significantly increases the probability to compromise circuits created by Tor clients. The proposed attack is active in the sense that the attacker tries to push a typical Tor client to open new connections through unpopular ports. Empirical analysis based on data obtained from the real Tor network showed that by injecting 30 pairs of entry/exit relays and redirecting the traffic through unpopular ports, the anonymity of Tor clients is reduced by more than 50%. Given the few number of Tor nodes allowing unpopular ports, this attack is still scalable with the current Tor network (January 2013). The main requirement for this attack to work is a Tor client that uses an unpopular application (port) through the Tor network. Table I lists the unpopular ports in Tor. In the sequel we show how it is possible to actively push a typical Tor client (using typical ports) to open a connection through Tor using one of the unpopular ports.

Our plan for future work is to implement more efficient hacking techniques to actively redirect the Tor traffic through unpopular ports. One promising approach would be to carry out specially crafted cross-site scripting attacks to remotely execute malicious scripts.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, “Tor : the second-generation onion router,” in *Proceedings of the 13th Usenix Security Symposium*, August 2004.

[2] O. Berthold, H. Federrath, and S. Köpsell, “Web MIXes: A system for anonymous and unobservable Internet access,” in *Proceedings of Designing Privacy Enhancing Technologies*. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[3] The Tor Project, Inc, “Tor project,” ”http://www.torproject.org”.

[4] A. Chaabane, P. Manils, and M. Kaafar, “Digging into anonymous traffic: A deep analysis of the tor anonymizing network,” in *Proceedings of the 2010 Fourth International Conference on Network and System Security*, ser. NSS ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 167–174.

[5] K. Loesing, S. Murdoch, and R. Dingledine, “A case study on measuring statistical data in the tor anonymity network,” in *Proceedings of the 14th international conference on Financial cryptograpy and data security*, ser. FC’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 203–215.

[6] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, “Shining light in dark places: Understanding the tor network,” in *Proceedings of the 8th international symposium on Privacy Enhancing Technologies*, ser. PETS ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 63–76.

[7] K. Bauer, D. Grunwald, and D. Sicker, “Predicting tor path compromise by exit port,” in *Proceedings of the 2nd IEEE International Workshop on Information and Data Assurance (WIDA)*, Phoenix, USA, December 2009.

[8] S. J. Murdoch and R. N. Watson, “Metrics for security and performance in low-latency anonymity systems,” in *Proceedings of the 8th international symposium on Privacy Enhancing Technologies*, ser. PETS ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 115–132.

[9] T. G. Abbott, K. J. Lai, M. R. Lieberman, and E. C. Price, “Browser-based attacks on tor,” in *Proceedings of the 7th international conference on Privacy enhancing technologies*, ser. PET’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 184–199.

[10] N. Evans, R. Dingledine, and C. Grothoff, “A practical congestion attack on tor using long paths,” in *Proceedings of the 18th USENIX Security Symposium*, August 2009.

[11] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “Low-resource routing attacks against Tor,” in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*, Washington, DC, USA, October 2007.

[12] Amazon Web Services, “Ec2: Elastic compute cloud,” ”aws.amazon.com/ec2”.

[13] I. Fette and A. Melnikov, “The websocket protocol (rfc 6455),” IETF Internet Draft, December 2011.

[14] R. Dingledine and N. Mathewson, “Tor path specification,” ”http://tor.eff.org/cvs/doc/path-spec.txt”.