Tor Bridge Discovery: Extensive Analysis and Large-scale Empirical Evaluation

Zhen Ling, Junzhou Luo, Wei Yu, Ming Yang, and Xinwen Fu

Abstract—Tor is a well-known low-latency anonymous communication system that is able to bypass the Internet censorship. However, publicly announced Tor routers are being blocked by various parties. To counter the censorship blocking, Tor introduced non-public *bridges* as the first-hop relay into its core network. In this paper, we investigated the effectiveness of two categories of bridge-discovery approaches: 1) enumerating bridges from bridge HTTPS and email servers, and 2) inferring bridges by malicious Tor middle routers. Large-scale real-world experiments were conducted and validated our theoretic findings. We discovered 2365 Tor bridges through the two enumeration approaches and 2369 bridges by only one Tor middle router in 14 days. Our study shows that the bridge discovery based on malicious middle routers is simple, efficient, and effective to discover bridges with little overhead. We also discussed issues related to bridge discovery and mechanisms to counter the malicious bridge discovery.

Index Terms—Anonymous communication, Tor, bridge discovery, attack, privacy

1 Introduction

Tor is a popular low-latency anonymous communication system and supports TCP applications over the Internet [1]. It has been commonly used for resisting the Internet censorship [2]. Because Tor uses source routing to achieve communication privacy and the information of all Tor routers is available to clients and publicly listed on the Internet [3], blocking Tor is as simple as blocking connections to those known Tor routers.

To resist the censorship blocking of public Tor routers, Tor introduced *bridges*. Generally speaking, a bridge could act as the first hop, which relays user traffic into the core Tor network. The bridge information remains hidden and is not listed on the Internet. There are a few bridge pools and some are stored at the bridge HTTPS and email servers. A user could access the bridge HTTPS server or send a google/yahoo email to the bridge email server to retrieve three bridges at one time. Bridges are distributed through various social networks as well.

Nevertheless, our study, along with other related research efforts [4], [5] have shown the two categories of bridge-discovery approaches: 1) the enumeration of bridges through bulk emails and Tor's HTTPS server, and 2) the use of malicious middle routers to discover bridges. Note that bridge may pick up malicious middle routers as the second hop of Tor routing path. Tor almost completely fails in some regions and we believe these regions may have blocked Tor bridges using these discovery approaches as well as blocking all public Tor routers. To this end, the censorship wins the battle against the user's privacy.

To fully understand the reason why Tor fails in some regions, we provide the *first formal analysis* and *large-scale* empirical evaluation of the effectiveness of Tor bridges resisting the censorship in this paper. We conduct an extensive theoretical analysis on two bridge-discovery approaches and our experimental results show the effectiveness of large-scale bridge discovery in real-world environments. To the best of our knowledge, although there are a few related research efforts on discovering bridges [4], [5], the discussion in those papers is very limited and there are no formal analysis and large scale real-world experiments as we conducted in this paper. The contributions of this paper are summarized below.

First, we formalize the bridge discovery through email and HTTPS enumeration as a weighted coupon collector problem and derive the expected number of bridges in terms of number of enumerations (samplings). Our real-world experiments support the theory well. In particular, we use a master machine to control over 500 PlanetLab nodes [6], by which emails are sent from 2000 yahoo email accounts in a round robin fashion. We also use a master machine to control over 1000 Tor and PlanetLab nodes, which send HTTPS requests to retrieve bridges from bridge HTTPS server. Our email and HTTPS enumeration approaches derived a list of 2365 Tor bridges around one month. Nevertheless, these two enumeration approaches incur considerable overhead. Yahoo and Google use *CAPTCHA* to prevent continuous generation of bulk email accounts. Tor takes countermeasures against malicious enumeration by controlling the number of fresh bridges obtained by an IP address or a subnet in a time duration. Therefore, the two enumeration approaches are not efficient and effective to some extent. In addition, we evaluate the impact of the bridge discovery and show how difficult the Tor bridge users could obtain the undiscovered bridges after a number of bridges have been discovered and blocked.

- *Z. Ling, J. Luo, and M. Yang are with Southeast University. E-mail: {zhenling, jluo, yangming2002}@seu.edu.cn.*
- *W. Yu is with Towson University. E-mail: wyu@towson.edu.*
- *X. Fu is with University of Massachusetts Lowell. E-mail: xinwenfu@cs.uml.edu.*

Manuscript received 12 Feb. 2013; revised 10 Sept. 2013; accepted 16 Sept. 2013. Date of publication 30 Sept. 2013; date of current version 05 June 2015. Recommended for acceptance by Y. Xiang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.249

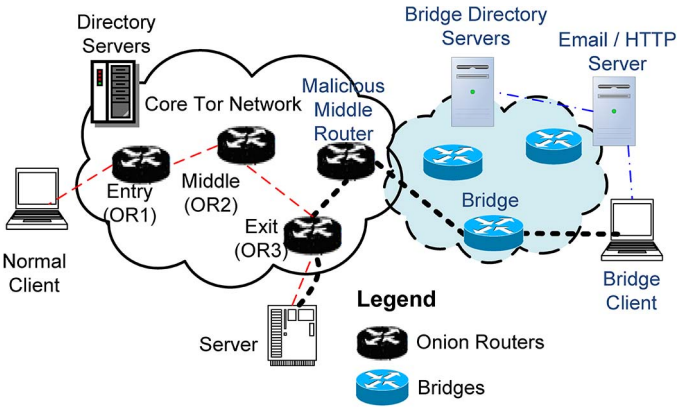


Fig. 1. Tor network.

Second, we formally analyze the capability of bridge discovery through malicious middle Tor routers. If a Tor router is not configured as an exit and does not meet the criteria of being an entry guard, it can only be a middle router. Hence, if a bridge's next hop is a malicious middle router, the middle router will find that the bridge's IP address that is not within the public Tor router list and thereby determine the bridge. Based on our real-world experiments, we could confirm that with three malicious middle routers that have a bandwidth of 10 MB/s, if 30 circuits are established through a bridge, the probability of discovering the bridge approaches 100 percent. In other words, if 30 clients use the same bridge to create a circuit, that bridge will be exposed in a probability of almost 100 percent. Our real-world experimental data show that the 21th circuit created by a bridge client traverses one of the 500 PlanetLab nodes **with bandwidth** 50 KB/s. Our analytical results show that the effectiveness of bridge discovery is actually determined by the total bandwidth contributed by those malicious middle routers, not just the number of malicious middle routers. Using one malicious middle router **with bandwidth** 10 MB/s in our experiments, we actually discovered 2369 Tor bridges over two weeks to validate our theory well. In summary, the malicious middle router based approach can discover bridges distributed by any approach and it is efficient and effective with little overhead. We also present the practical issues related to our bridge discovery and discuss the feasibility and effectiveness of potential countermeasures against bridge discovery.

The rest of the paper is organized as follows. In Section 2, we introduce the components of Tor and bridges along with the basic operation of Tor for both normal clients and bridge clients. In Section 3, we present our approaches for discovering Tor bridges through email, HTTPS, and Tor middle routers. In Section 4, we analyze the effectiveness of those bridge discovery approaches. In Section 5, we show the real-world experimental results on Tor and validate our theory. We discuss issues related to bridge discovery and present a set of guidelines to counteract those bridge discovery approaches in Section 7. We review related work in Section 6 and conclude the paper in Section 7, respectively.

2 Background

In this section, we first review the components of Tor and bridges and then present the basic operation of Tor from both normal clients's and bridge clients' aspects. Note that Tor algorithms presented in this paper were discovered from reading the Tor project source code and we show some details in this section, which are not provided by Tor documents. Such details make our analysis complete and match the real-world Tor behavior.

2.1 Basic Components of Tor and Bridge

Fig. 1 illustrates basic components of Tor with bridges. The *client* runs a local software denoted as *onion proxy (OP)* to anonymize the client data into Tor. We differentiate two types of clients: 1) *normal clients* use the Tor core network directly, and 2) *bridge clients* use bridges to access the Tor core network. The *server* runs applications such as web service and anonymously communicates with the client over Tor. *Onion routers (OR)* (or Tor routers) relay the application data between clients and servers. *Directory servers* hold the onion router information such as IP address. All users have a copy of onion router list locally. This is the main reason why it is easy to block the Tor core network.

Functions of onion proxy, onion router, directory servers, and bridge are integrated into the Tor software package. A user may edit the configuration file to configure a computer to have different combinations of those functions. Bridges are special Tor routers. Bridges publish their information to a bridge directory server and this server holds the information of all bridges. A client can retrieve the bridge information by accessing the bridge HTTPS or email server or obtain it from social networks directly.

2.2 Normal Clients Using Tor

To anonymously communicate with a web server, a normal client uses source routing and chooses a series of onion routers from the locally cached directory [7]. We denote the series of onion routers as a *path* through Tor [8], along which a circuit will be setup incrementally. The number of routers is denoted as the path length. **The detail of path selection process can be found in Algorithm 1 listed in Appendix A of supplemental file** which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.249. Note that Tor does not choose the same router twice for the same path [8]. From Algorithm 1, we can see that to create a circuit, Tor selects an exit node, an entry node and then a middle node in order. There are four types of Tor routers:

1. pure entry router (entry guard),
2. pure exit router,
3. both entry and exit router (denoted as *EE* router), and
4. neither entry nor exit router (denoted as *N-EE* router).

A router is marked as an entry guard by the authoritative directory server only if its mean uptime is above the median of all ''familiar'' routers and its bandwidth is greater than $\max(median, 250$ KB/s) [7]. A router is designated as an exit only if it allows traffic to go out to two public ports among 80, 443, and 6667, and at least one

class C IP address space. An EE router is one marked as both entry guard and exit router by directory servers, and N-EE router is marked as neither of them.

To ensure the performance, Tor adopts the weighted bandwidth routing algorithms. In the following, we use the default path length of 3 in Fig. 1 to illustrate how a path is selected for the normal clients.

First, the client chooses an appropriate exit onion router $OR3$ from the set of exit routers, including the pure exit routers and EE routers. **The node exclusion process for selecting exit nodes can be found in Algorithm 2 listed in Appendix A of supplemental file available online. Note that this algorithm excludes nodes that are not running or remarked as bad exits, not meeting capacity or uptime requirements, remarked as invalid, or have policies that reject all traffic**. The bandwidth of exit routers is weighted and described below. Assume that the total bandwidth is B, the total exit bandwidth is B_E, and the total entry bandwidth is B_G. If $B_E < \frac{B}{3}$ (i.e., the bandwidth of exit routers is scarce), the exit routers will not be considered for non-exit use. The bandwidth of EE routers are weighted by

$$W_G = 1 - \frac{B}{3B_G}, \tag{1}$$

where W_G is the bandwidth weight of entry routers and $B_G > \frac{B}{3}$. If $B_G < \frac{B}{3}$, we have $W_G = 0$. The probability of selecting the ith exit router from the exit set is $\frac{Bi_E}{B_{exit}+B_{EE}\cdot W_G}$, where B_{EE} is the total bandwidth of EE routers, B_{exit} is the total bandwidth of pure exit routers, and Bi_E indicates the bandwidth of ith exit router. Note that $B_E = B_{exit} + B_{EE}$. **Algorithm 3 in Appendix A of supplemental file available online depicts the detailed bandwidth weighted node selection procedure, which is used to select the exit node from the corresponding candidates**.

Second, the client chooses an appropriate entry onion router $OR1$ from the set of entry routers, including the pure entry routers and EE routers. Denote the total bandwidth of pure entry routers as B_{entry}, where $B_G = B_{entry} + B_{EE}$. To ensure sufficient entry bandwidth, if $B_G < \frac{B}{3}$, the entry routers will not be considered for non-entry use. Then, the probability of selecting the ith entry router from the entry set is $\frac{Bi_G}{B_{entry}+B_{EE}\cdot W_E}$, where $W_E = 1 - \frac{B}{3B_E}$. Here W_E is the exit bandwidth weight and Bi_G is ith bandwidth in the entry set. If $B_E < \frac{B}{3}$, we have $W_E = 0$. Algorithm 3 is also used to choose the entry node from the corresponding candidates.

Third, any router can be selected as the middle onion router $OR2$ except the already selected routers $OR3$ and $OR1$. **Algorithm 4 in Appendix A of supplemental file available online describes the selection of middle nodes for a circuit**.[1] In Algorithm 3, bandwidth is required for creating a circuit by default. Hence, the bandwidth weighted router selection algorithm is used to select a middle router. The probability of selecting ith router from the remaining set of Tor routers is $\frac{Bi}{B_{exit}\cdot W_E+B_{entry}\cdot W_E+B_{EE}\cdot W_E\cdot W_G+B_{N-EE}}$, where Bi is ith bandwidth in the remaining set and B_{N-EE} is the total bandwidth of N-EE routers.

1. Note that another constraint is that each router on a path must be selected from a distinct/16 subnet.

The bandwidth weighted node selection algorithm, i.e., Algorithm 3, not only improves the load balance of the entire Tor network, but also improves the end-to-end performance for Tor clients. If the available total bandwidth of exit routers is scarce in the Tor network, e.g., $B_E < \frac{B}{3}$, the probability that exit Tor nodes are selected for entry and middle positions is smaller by using such bandwidth weighted algorithm. In addition, all of the EE nodes will be chosen as exit routers to increase the total bandwidth of exit routers. In this way, the bandwidth weighted node selection algorithm can balance the bandwidth over the entire Tor network. In addition, this algorithm can increase the probability that the high-bandwidth Tor routers will be selected in the path. Thus, from the client's point of view, it can improve the performance of anonymous communication path in the Tor network.

Once the path is chosen, the client creates a circuit over the path incrementally, one hop at a time. A circuit is a communication tunnel encrypted in an onion-like way over the path. Once the circuit is established, the client can connect to a web server through the circuit.

2.3 Bridge Clients Using Tor

In a censored region, all public Tor routers can be blocked. To access the core Tor network, Tor clients may utilize non-public Tor routers, called as bridges. A bridge client needs to obtain at least one bridge. As we can see from Fig. 1, the bridge client can obtain the information of bridges through email and HTTPS. We will further discuss these approaches in Section 3. The bridge client uses a bridge as a hidden first-hop relay into the Tor network to avoid censorship. The bridge client then follows the similar procedures discussed earlier, i.e., downloading the information of Tor nodes and choosing the appropriate exit onion router $OR3$ and middle onion router (e.g., malicious middle router in Fig. 1), respectively. Finally, the bridge client creates a circuit and anonymously surfs the Internet.

We would like to note that malicious middle routers cannot directly help an adversary to identify a user's current IP address. Tor introduces non-public bridges as the first-hop relay, replacing the entry node. Along a circuit, the bridge knows IP addresses of incoming cells and those connecting clients, the exit router knows the destination IP address of a cell, and the middle router only knows IP address of a bridge and the exit router. Therefore, even if one of these three routers is compromised, the adversary can not compromise the communication privacy of bridge clients. In this paper, we investigate how the adversary controls middle routers to discover Tor bridges.

3 Three Approaches for Large-Scale Tor Bridge Discovery

In this section, we first introduce the basic ideas of discovering Tor bridges and then present our experimental strategies.

3.1 Basic Ideas

In this paper, we focus on the following two categories of approaches to discover bridges:

1. *Email and https enumeration*. An adversary can use a Yahoo email or gmail account to send an email to the

bridge email server (*bridges@torproject.org*) with the line ''get bridges'' in the body of the mail. The bridge email server promptly replies with three distinct bridges. To avoid malicious enumeration, the bridge email server only replies one email to an email account each day. Alternatively, the user can access the bridge website (https://bridges.torproject.org/) to obtain three bridges. To avoid malicious enumeration, the https server distributes three bridges to each 24-bit IP prefix each day as well.

2. *Bridges inference by malicious Tor middle routers*. A circuit created by a bridge client traverses both the bridge and the malicious middle router. By deploying middle routers in home, PlanetLab or Amazon EC2, we may discover bridges from any bridge pool, including those privately distributed in social networks.

3.2 Discovering Bridges via Email

We can enumerate Tor bridges through a massive number of email accounts. The Tor bridge email server only replies to Yahoo email and gmail. To obtain the 2000 yahoo email accounts, we use *iMacros* [9] to automate the email account application. *iMacros* can record email application procedures into a script and repeat most of the work automatically. During each automation cycle, humans still need to change the email account, fill out the CAPTCHA, and submit the application. Yahoo limits the number of email account applications from a single IP address. To address this issue, we deploy more than 500 PlanetLab nodes to carry out the email application tasks each day. We may also use the Tor network to apply for email accounts. More than 500 Tor exit routers were used as the proxies [10]. Consequently, those exit routers provide enough distinct IP addresses for acquiring a large number of email accounts. PlanetLab nodes can also be used as proxies for email account application.

We adopt a command-and-control architecture to send bulk emails soliciting bridges. Yahoo does not allow a large number of emails from a single IP address via SMTP (Simple Mail Transfer Protocol). We use a master computer to control the PlanetLab agents, which are deployed to the PlanetLab nodes through a parallel SSH execution tool *Pssh* [11]. Agents receive the email accounts and passwords from the master server and send emails to the bridge email server. A tiny SMTP client [12] is used by a PlanetLab agent. Because Yahoo does not provide free *POP3* (Post Office Protocol 3) service, we use a tiny *POP3* client *Mpop* [13] to retrieve Yahoo emails through an emulated POP3 server *FreePOPs* [14], which is able to access Yahoo webmail service. A script can then be used to analyze the downloaded emails and retrieve the IP addresses of bridges embedded in emails.

To the best of our knowledge, this is the first time that bulk emails are used for enumerating Tor bridges.

3.3 Discovering Bridges via HTTPS

Because Tor limits bridge retrieval from each class *C* IP address, we have to control a large number of hosts with different IP address prefixes to obtain a large number of bridges within a short time. To this end, we introduce the following two schemes:

1. *https via PlanetLab nodes*. A master computer can control a large number of PlanetLab nodes for retrieving the bridges. We select around 500 PlanetLab nodes and upload the agent software to each node. An agent receives the command from the master to download the bridge webpage via *wget*. To avoid congesting the Tor https server, the master manages the PlanetLab agents in a round robin fashion for bridge retrieval. We use the parallel SSH execution tool *Pslurp* to download the webpages from the PlanetLab agents and a script is used to analyze the webpages for embedded bridges.
2. *https via Tor exit nodes*. Tor has around 500 exit nodes. Most of them have IP addresses with different 24-bit IP prefixes. We use a Tor client to create the circuits through different exit nodes and retrieve bridges via https. We implemented this approach by exploring the Tor control protocol [15], which is an interface between the customer programs and the Tor network. The control protocol allows a client to control its usage of Tor and acquire Tor status, including circuit status and others. Therefore, a malicious Tor client can utilize this control protocol to command Tor. The control messages include ''command,'' ''keyword,'' and ''arguments''. There are different ''commands'' for various functionalities. For example, the command ''SETCONF'' changes the values of Tor configuration variables. We can utilize the command ''EXTENDCIRCUIT'' to establish a new circuit along a specified path.

Tor has a cross-platform controller GUI, *Vidalia*[16]. We implemented the custom circuit creation using Tor control protocol and integrated it into *Vidalia*. To successfully create and use custom circuits, we should first disable Tor's automatic circuit creation mechanism by using two commands, ''SETCONF __DisablePredictedCircuits=1'' and ''SETCONF MaxOnionsPending=0''. We then use the command ''EXTENDCIRCUIT CircuitID ServerSpec *(, ServerSpec)'' to establish our custom circuits. If ''CircuitID'' is zero, it is a request that Tor build a new circuit along the specified path. Otherwise, it is a request that the server should extend an existing circuit with that ID along the specified path. Note that ''ServerSpec'' is the nickname of the specified Tor node. In this way, we can control the Tor network to create a two-hop circuit through distinct exit nodes. Once the circuit is created, the tool *wget* is used to download bridge web pages.

3.4 Discovering Bridges through Tor Middle Routers

Fig. 1 illustrates the basic idea of discovering Tor bridges through middle Tor routers. We deploy malicious Tor middle routers on PlanetLab to discover bridges connected to these Tor middle routers. Recall that a client uses a bridge as an entry node to establish a three-hop circuit for surfing the Internet. Traffic forwarded by the bridges may traverse these middle routers. Then the middle routers can identify the IP addresses of bridges. The recorded IP addresses will be either from Tor entry guards or from bridges. Because the information of entry guards is public,

it is trivial to distinguish bridges from entry nodes. We modified the Tor source code to embed the aforementioned functions, record the incoming connection information, differentiate bridges from other Tor nodes, and send an email with the IP addresses of bridges to us. This approach allows us to automatically retrieve bridges through the controlled Tor middle routers on PlanetLab. Of course, such malicious middle routers can be deployed at any place, including the researchers' home and Amazon EC2 [17]. PlanetLab nodes have very limited bandwidth while home and Amazon EC2 nodes may provide large bandwidth.

Note that we need to prevent malicious routers from becoming entry or exit routers automatically because of the rule of Tor. When onion routers advertise an uptime and bandwidth at or above the median among all routers, these routers will be marked as entry guards by directory servers [7]. To prevent malicious routers from being entry routers, we need to reduce their bandwidth or control their uptime. By configuring the exit policy, we also prevent those malicious routers from being exit routers.

4 Analysis

In this section, we first analyze the effectiveness of the bridge discovery through emails and HTTPS. To this end, we formalize the bridge discovery problem as a weighted coupon collector problem and derive the expected number of samplings for obtaining all bridges. We then analyze the effectiveness of the bridge discovery approach through malicious Tor middle routers.

4.1 Bridge Discovery through Emails and HTTPS

The effectiveness of Tor bridge discovery approaches through email and HTTPS presented in Section 3 can be measured by the number of derived bridges over time. We can formalize the process of Tor bridge discovery through emails and HTTPS as a weighted coupon collector problem. In the typical coupon collector problem [18], all m coupons are obtained with an equal probability with replacement and one of these coupons is drawn in each sampling. The expected time for a collector to derive the m distinct coupons is $\sum_{i=1}^{m} \frac{m}{m-i+1} = m \sum_{i=1}^{m} \frac{1}{i} \approx \Theta(m \ln m)$. Nevertheless, in our case, bridges are not distributed with equal probability, but weighted based on bandwidth as we discussed in Section 2, which is also validated by our real-world experiments in Section 5.

We now derive the weight for our weighted coupon collector problem. To enhance performance, Tor adopts the weighted bandwidth routing algorithm for the path (circuit) selection. The higher a router's advertised bandwidth, the higher the chance that the router will be selected for a circuit. Note that a bridge can act as a Tor entry router. With such a weighted bandwidth routing algorithm, we assume that the bandwidth of the bridges comprises a set $\{B_1, B_2, \ldots, B_n\}$, where n is the number of Tor bridges in the Tor network. The probability p_i that the ith router with bandwidth B_i can be selected is

$$p_i = \frac{B_i}{\sum_{i=1}^{n} B_i}. \tag{2}$$

Based on the Tor bridge discovery described in Section 3, we now give the problem definition of our weighted coupon collection problem for discovering bridges. *Given n bridges, the ith bridge can be derived in each sampling with the probability p_i, where $0 < p_i < 1$.* What is the expected number of samplings to derive n distinct bridges?

According to the weighted coupon collector's problem, we can derive the expected samplings to derive the n distinct bridges through email and HTTPS. We denote I_i as a random variable and define

$$I_i = \begin{cases} 1, & \text{if the } i\text{th bridge is collected during } h \text{ samplings} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Denote X_h as the number of distinct bridges after $h(\geq 1)$ times of bridge retrieval through email or HTTPS. The probability that the ith bridge is not collected after h samplings is $P(I_i = 0) = (1 - p_i)^h$. Then, the number of distinct bridges after h samplings becomes $X_h = \sum_{i=1}^{n} I_i$. Because we have

$$\begin{aligned} E(I_i) &= 0 \times P(I_i = 0) + 1 \times P(I_i = 1), \\ &= P(I_i = 1) = 1 - P(I_i = 0) = 1 - (1 - p_i)^h, \end{aligned} \tag{4}$$

the expected number of different bridges discovered after h samplings can be derived by

$$\begin{aligned} E(X_h) &= E\left(\sum_{i=1}^{n} I_i\right), \\ &= E(I_1) + E(I_2) + \cdots + E(I_n), \\ &= 1 - (1 - p_1)^h + \cdots + 1 - (1 - p_n)^h, \\ &= n - \sum_{i=1}^{n} (1 - p_i)^h. \end{aligned} \tag{5}$$

It can be observed from Equation (5) that when we conduct more samples (i.e., h increases), we can obtain more bridges. Note that to avoid malicious enumeration, the bridge authority divides the available bridges into pools. Each pool is available in a time window [19]. However, in one time window, the enumeration can still be formalized as a weighted coupon collector problem. This is confirmed by our experimental results shown in Fig. 4.

4.2 Bridge Discovery through Middle Routers

Recall that if a TCP stream from a bridge traverses malicious Tor middle routers, the bridge will be exposed. To understand the effectiveness of this bridge discovery approach, we analyze the **catch probability** that a TCP stream from a bridge traverses malicious middle routers.

We assume that k computers are injected into the Tor network and there are malicious Tor middle routers. The bandwidth of all onion routers comprises a set $\{B_1, B_2, \ldots, B_k, B_{k+1}, \ldots, B_{k+N}\}$, where $\{B_1, \ldots, B_k\}$ is the bandwidth of the malicious middle routers. All malicious middle routers advertise the same bandwidth[2], $B_1 = B_2 = \cdots = B_k = b$. Denote B as the aggregated bandwidth

2. The Tor project released a new version that changes the upper-bound of high bandwidth to 10 MB/s on August 30, 2007.

of all original onion routers, $B = \sum_{i=k+1}^{k+N} B_i$. Then the total bandwidth becomes $B + k \cdot b$.

Recall that there are four types of routers in the Tor network:

1. *pure entry router* (entry guard),
2. *pure exit router*,
3. *both entry and exit* router (denoted as *EE* router), and
4. *neither entry nor exit* router (denoted as *N-EE* router).

Denote the bandwidth of all original pure entry routers, pure exit routers, EE routers and N-EE routers as B_{entry}, B_{exit}, B_{EE} and B_{N-EE}, respectively. Note that we have $B = B_{entry} + B_{N-EE} + B_{EE} + B_{exit}$. Based on the weighted bandwidth routing algorithm discussed in Section 2.2, the bandwidth weight can be derived by,

$$W_E = \begin{cases} 1 - \frac{B+k\cdot b}{3\cdot(B_{exit}+B_{EE})} : & W_E > 0, \\ 0 : & W_E \leqslant 0. \end{cases} \quad (6)$$

$$W_G = \begin{cases} 1 - \frac{B+k\cdot b}{3\cdot(B_{entry}+B_{EE})} : & W_G > 0, \\ 0 : & W_G \leqslant 0. \end{cases} \quad (7)$$

The weighted bandwidth $B_{exit'}$, $B_{EE'}$, $B_{entry'}$, and $B_{N-EE'}$ can be derived as follows, $B_{exit'} = B_{exit} \cdot W_E$, $B_{EE'} = B_{EE} \cdot W_E \cdot W_G$, $B_{entry'} = B_{entry} \cdot W_G$, and $B_{N-EE'} = B_{N-EE} + k \cdot b$.

With the total weighted bandwidth $B_{exit'} + B_{EE'} + B_{entry'} + B_{N-EE'}$ derived above and the total bandwidth of malicious Tor middle routers $k \cdot b$, according to the weighted bandwidth route selection algorithm in Section 2.2 (i.e., the total bandwidth of malicious Tor middle routers divided by the total weighted bandwidth is the probability that malicious middle nodes are chosen for serving circuit), we have the following theorem to calculate the catch probability.

Theorem 1. *The* ***catch probability*** *can be derived by*

$$P(k, b) = \frac{k \cdot b}{B_{exit'} + B_{EE'} + B_{entry'} + B_{N-EE'}}, \quad (8)$$

where $k = 1, 2, 3 \ldots$ *and* $0 < b < 10$ MB/s.

Theorem 1 is intuitive based on the bandwidth weighted path selection algorithm. From Theorem 1, we derive the following corollaries.

Corollary 1. *The catch probability increases with the number of malicious Tor middle routers.*

$$P(r, b) > P(k, b), \quad \text{where} \quad r > k. \quad (9)$$

Corollary 2. *The catch probability increases with the bandwidth of malicious Tor middle routers, i.e.,* $P(k, b)$ *is a monotonous increasing function in terms of* b*. That is,* $P(k, l) > P(k, b)$*, where* $l > b$*.*

The proof of Corollary 1 and Corollary 2 is given in Appendix B and Appendix C of supplemental file available online respectively. These two corollaries indicate that the catch probability increases with both the number of malicious Tor middle routers and the bandwidth of malicious Tor middle routers. This matches with our expectation.

We would like to know what affects the catch probability, the number of malicious middle routers or the aggregated bandwidth of malicious middle routers. This is important in practice because we may not have so many computers with different IP addresses. Theorem 2 answers this question.

Theorem 2. *The catch probability is determined by the aggregated bandwidth contributed by malicious Tor middle routers. That is, if* $M = k \cdot b$*,* $M' = k' \cdot b'$*, and* $M \geq M'$*,* $P(M) \geq P(M')$*. The equality holds when* $M = M'$*.*

The proof of Theorem 2 is given in Appendix D. Theorem 2 implies that an adversary may not need to inject many malicious middle routers into the Tor network. A middle router with large bandwidth can achieve the same catch probability as a number of middle routers with small bandwidth. Our experiments in Section 5.2 validate this observation.

In practice, we also want to know the relationship between the catch probability and number of created circuits. Theorem 3 answers this question.

Theorem 3. *After* q *circuits are created, the catch probability that at least one bridge connects to one of the malicious* k *routers with bandwidth* b can be derived by

$$P(k, b, q) = 1 - (1 - P(k, b))^q, \quad \text{where} \quad q = 1, 2, 3, \ldots. \quad (10)$$

Theorem 3 is intuitive. From Theorem 3, we have Corollary 3.

Corollary 3. *The catch probability increases with the number of created circuits*

$$P(k, b, h) > P(k, b, q), \quad \text{where} \quad h > q. \quad (11)$$

The proof of Corollary 3 is given in Appendix D.

We also derive the relationship among the catch probability, the total bandwidth of the malicious Tor middle routers, and the number of created circuits based on Equation (10).

Corollary 4. *After* q *circuits are created, the probability that at least one bridge connects to one of the malicious routers with the total bandwidth of the Tor nodes* M *can be derived by,*

$$P(M, q) = 1 - (1 - P(M))^q. \quad (12)$$

Corollary 4 is intuitive given Theorem 3. From Theorems 2 and 3, we also derive Corollary 5. The detailed proof is given in Appendix D.

Corollary 5. *The catch probability increases with the total bandwidth of the malicious Tor middle routers.*

$$P(M, q) > P(M', q), \quad \text{where} \quad M > M'. \quad (13)$$

In summary, the catch probability increases with two factors: 1) the total bandwidth of malicious Tor middle routers, and 2) the number of created circuits. Our experimental data in Section 5 validate these theoretical

findings well. Our theoretical findings and experimental data show that the bridge discovery approach through Tor middle routers is feasible and effective even if we can only control a small number of Tor middle routers.

5 Evaluation

We have implemented the proposed Tor bridge discovery approaches in Section 3. In this section, we present the results of the large-scale empirical evaluation on these approaches. Our experimental results match the theoretical analysis presented in Section 4 well.

5.1 Bridge Enumeration through Email and HTTPS

To evaluate the bridge-discovery approaches through emails and HTTPS, we conducted large-scale experiments on PlanetLab from May to June 2010. Fig. 2 shows the number of newly collected distinct bridges, the number of totally collected distinct bridges, and the number of collected emails over the time. Because the Yahoo SMTP server may not successfully deliver emails sent from PlanetLab, we could not receive all replies all the time. From Fig. 2, we can see that more emails produce more distinct bridges. On May 5, 2010, we received 2000 emails and collected more bridges than other dates.

Fig. 2 also shows that the number of totally collected bridges increases over time. Actually, we are told that Tor has more than 10,000 bridges! This is the reason why the number of bridges keeps increasing steadily. However, this set of experiments show that the discovery approach works effectively because it could continually discover the new bridges. To enumerate all bridges, we only need to continue experiments. Fig. 3 shows the data obtained through HTTPS. Note that the number of discovered bridges through HTTPS has a similar trend to the one in Fig. 2.

We now verify Equation (5) in Section 4 using the real-world data. Recall that Tor distributes different pools of bridges (there is crossover among pools) through email and HTTPS servers over time. However, experiments on one day can be formulated as a weighted coupon collector problem because the pool has not been shifted. One retrieved bridge can be treated as one sampling. Fig. 4 shows the relationship

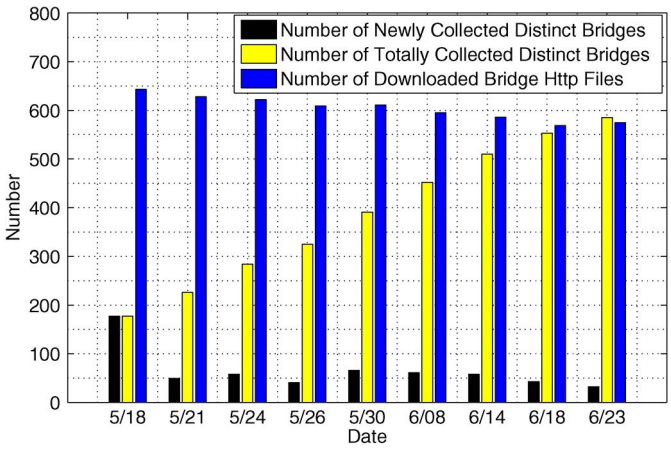


Fig. 3. Discovered bridges through HTTPS.

between the number of samplings and the number of distinct bridges. It can be observed that the curve of the non-weighted classical coupon collector problem is much steeper than the curve for the real-world data at the beginning. This indicates that the bridges are not uniformly distributed.

To verify that the bridge distribution can be formalized as a weighted coupon collector problem, we assume that the bridge bandwidth distribution is similar to the public Tor router bandwidth distribution. We randomly pick up a set of public Tor routers and use their bandwidth to simulate the bridge bandwidth (note that we do not know the bridge bandwidth). We then obtain the curve of the weighted coupon collector problem based on Equation (5). Fig. 4 shows that the theoretical curve is only slightly lower than the curve from the real-world data. Hence, it is highly possible that bridges are distributed with their bandwidth as the weight. Such a weighted distribution is also consistent with the Tor's weighted routing algorithm for performance enhancement. Actually, Tor developers later confirmed this fact to us.

5.2 Bridge Discovery via Tor Middle Routers

Fig. 5 shows the public Tor router bandwidth cumulative distribution function on July 10, 2010. Based on the data

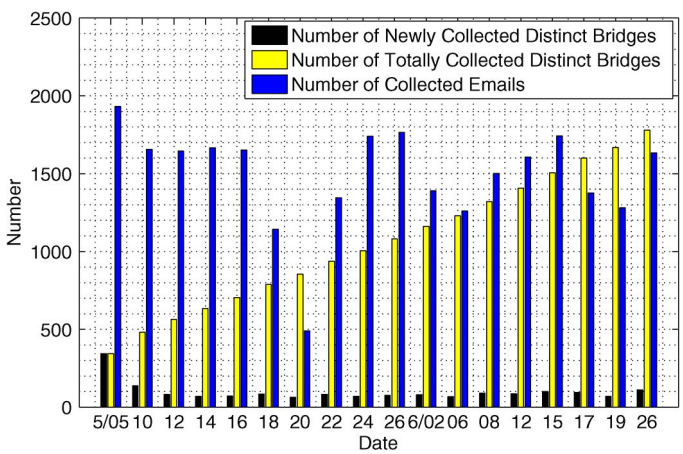


Fig. 2. Discovered bridges through emails.

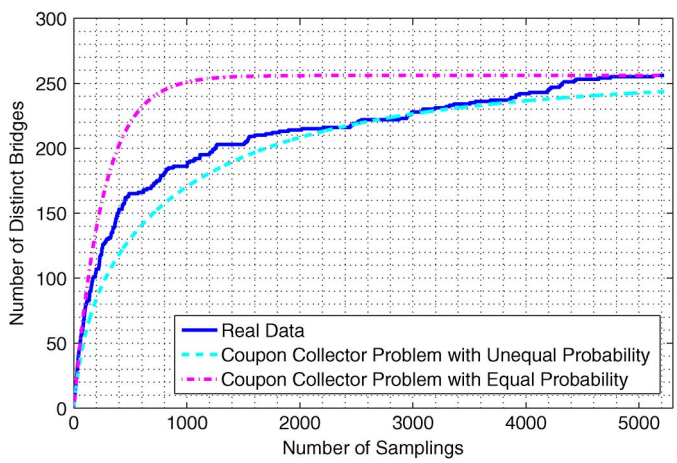


Fig. 4. Number of samplings vs. number of distinct bridges.

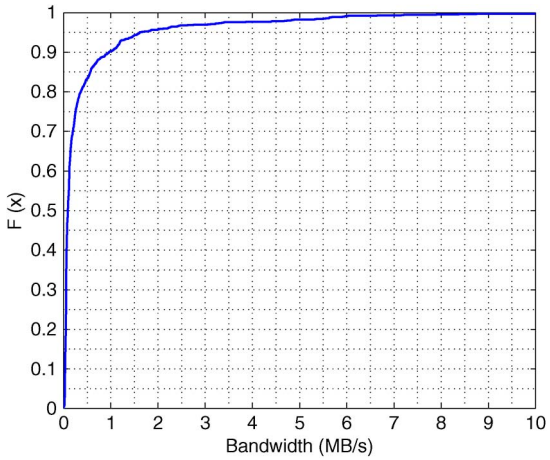


Fig. 5. Empirical CDF of bandwidth of all routers in the tor network.

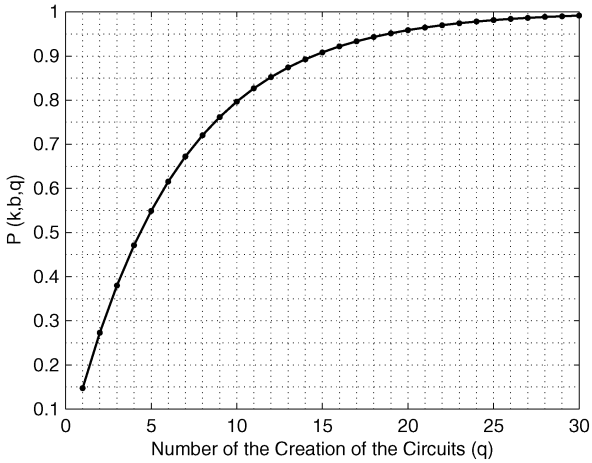


Fig. 7. Probability that at least a circuit traverses through the controlled middle routers after bridge clients create q circuits.

from Tor Metric Portal [20], there were 1604 active Tor routers, including 326 pure entry onion routers, 525 pure exit onion routers, and 132 EE routers. Using real-world data, Fig. 6 shows the relationship between the theoretical catch probability and the number of controlled Tor middle routers based on Theorem 1. As we can see, the catch probability is 14.7 percent when 512 Tor middle routers with bandwidth 50 KB/s are used, i.e., $P(512, 50) = 14.7\%$. Based on Theorem 3, Fig. 7 illustrates the catch probability when the bridge clients create q circuits, that is $P(512, 50, q)$. From Fig. 7, we can see that in theory, the catch probability approaches 99 percent, after the bridge clients created 30 circuits, i.e., $P(512, 50, 30) \approx 99\%$. In addition, from Equation (13), we know that by only configuring three nodes as malicious Tor middle routers, we can obtain the catch probability $P(3 * 10000, 30) > P(512 * 50, 30) \approx 99\%$.

To demonstrate the correctness of our theory, we used 512 PlanetLab nodes as malicious Tor middle routers and set their bandwidth as 50 KB/s (because of the limited bandwidth on PlanetLab nodes). To avoid affecting the Tor network, we only conducted a short experiment for two days. We set up a Tor client to create 430 circuits via our own Tor bridge in an apartment. We found that the 21th circuit passed through our Tor middle routers on PlanetLab. The experimental results match the theoretical results above well.

We now show data supporting the fact that high bandwidth routers have a higher chance to be selected as middle routers. Fig. 8 shows the empirical cumulative distribution function (ECDF) of the bandwidth of onion routers, which are selected as middle routers for these 430 circuits. Recall that we are able to record routers selected for a circuit at the client. We can see that 60 percent of those routers have a bandwidth more than 1 MB/s. However, as shown in Fig. 5, only 10 percent of Tor routers have a bandwidth of larger than 1 MB/s. This indicates that the higher bandwidth the routers have, the higher chance these routers are selected as middle routers for serving circuits.

Fig. 9 illustrates the theoretical catch probability that a circuit passes malicious Tor middle routers in terms of router bandwidth advertisement and the number of malicious middle routers, based on Theorem 1. We can see that the theoretical probability is monotonically increased with both the number of controlled middle routers and their bandwidth advertisement. These observations match our analytical results in Theorems 1 and 2

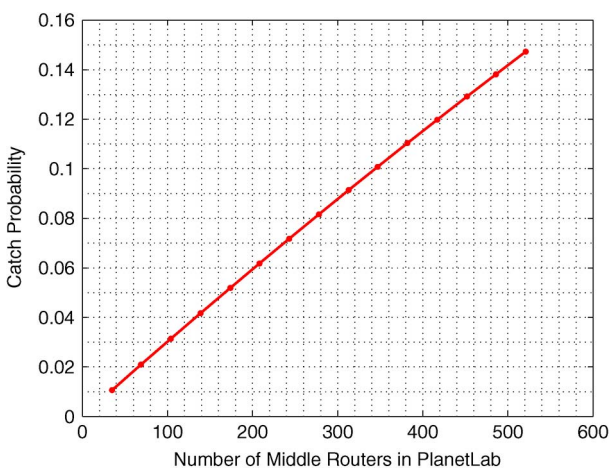


Fig. 6. Probability that a circuit chooses the middle routers in PlanetLab vs. number of Tor middle routers in PlanetLab.

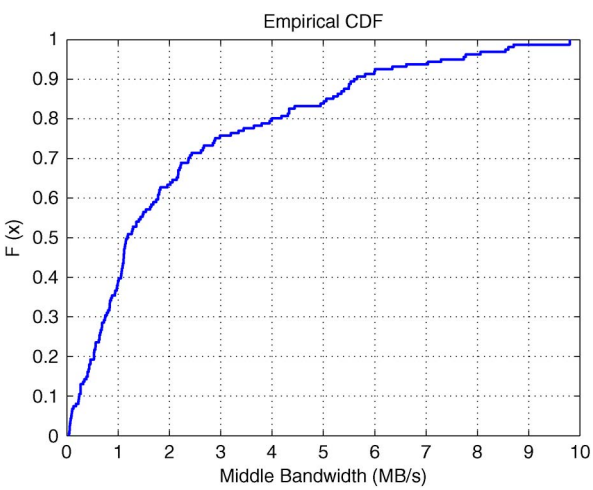


Fig. 8. Empirical CDF of bandwidth of selected middle routers.

well. As expected, the catch probability approaches 90 percent when there are 20 malicious middle routers with 10 MB/s bandwidth, i.e., $P(20, 10000) \approx 90\%$.

To validate Theorem 2 that the bridge discovery is determined by the aggregated bandwidth of the malicious Tor nodes, we configured a high bandwidth middle router **with bandwidth** 10 MB/s. We recorded the bridges that pass through this middle router from July 10 to 23, 2010. Fig. 10 shows the number of newly discovered distinct bridges and the number of totally collected distinct bridges. As we can see, the number of totally collected bridges increases over time. Eventually, we obtained 2369 bridges, indicating that discovering bridges through middle routers can be very effective and efficient and the catch probability is mainly determined by total bandwidth contributed by malicious middle routers. Note that to prevent the middle router from becoming an entry router in 7 days, we restarted the router on the 6th day.

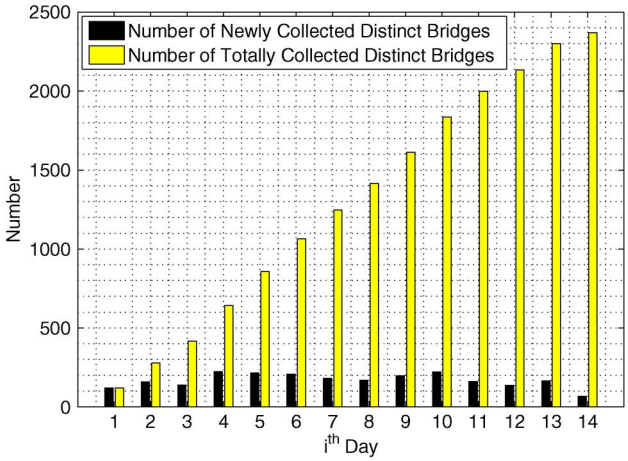


Fig. 10. Discovery of bridges through a Tor middle router.

5.3 Impact of Bridge Discovery through Email and HTTPS

We evaluated the impact of the large-scale bridge discovery and show how difficult the Tor bridge users obtain the undiscovered bridges after a number of bridges have been discovered and blocked. According to our experimental results in Fig. 4, we consider that the bridge distribution through email and HTTPS is based on bandwidth. Thus, the adversary will first discover the high bandwidth bridges through email and HTTPS and block them. Assume that the top p percentage bridges are discovered and blocked. We denote the bandwidth of the top p percentage bridges and the total bandwidth of the Tor routers as B'_p and B', respectively. Hence, the probability that Tor bridge users derive the top p percentage bridges is $\frac{B'_p}{B'}$. Then, we know that the probability P_p that the Tor bridge users derive at least one undiscovered bridge each time is $1 - \left(\frac{B'_p}{B'}\right)^3$. Recall that the bridge authority will distribute three bridges each time. We use the published bandwidth information of existing public Tor routers to simulate bridge bandwidth and sort the bandwidth of the Tor routers by descent. Fig. 11 depicts the probability that the Tor bridge users derive at least one undiscovered bridge each time after top p percentage bridges have been discovered and blocked. As we can see from this figure, if top 90 percent bridges are discovered, P_p is 22.8 percent. If 90 percent bridges are discovered and blocked, the probability $P_{90}(\ell)$ that the Tor bridge users try ℓ times to derive at least one unblocked bridge is $P_{90}(\ell) = 1 - \left(\frac{B'_p}{B'}\right)^{3\ell}$. Fig. 12 illustrates the relationship between $P_{90}(\ell)$ and the number of samplings. As we can see that the users need to try around 20 times to obtain at least one unblocked bridge. Note that Tor has changed the bridge distribution strategy to distribute three bridges each week instead of each day. Hence, it will take around 20 weeks to get at least one unblocked bridge via email or HTTPS. We believe it will be a fairly uncomfortable experience for the Tor bridge users.

5.4 Effectiveness of Bridge Discovery through Controlled Middle Routers

We also evaluated the effectiveness of the bridge discovery through controlled middle routers and show how effective

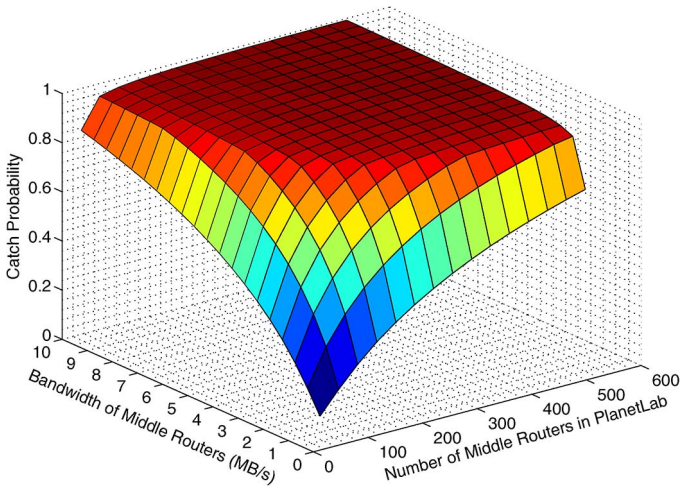


Fig. 9. Probability that a circuit chooses the Tor middle routers in PlanetLab vs. Number of Tor middle routers & bandwidth advertisement of Tor middle routers in PlanetLab.

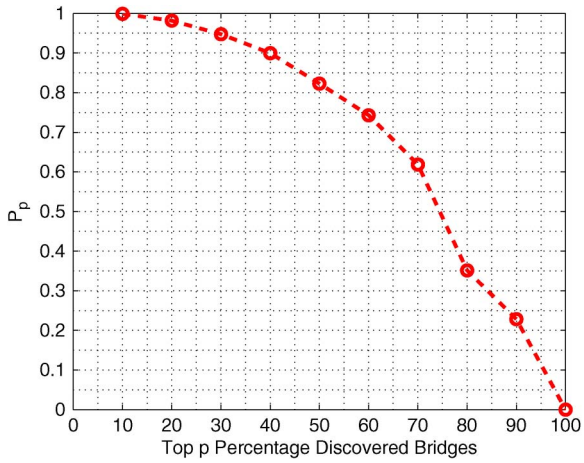


Fig. 11. Probability that Tor bridge users derive at least one undiscovered bridge each time after top p percentage bridges have been discovered and blocked.

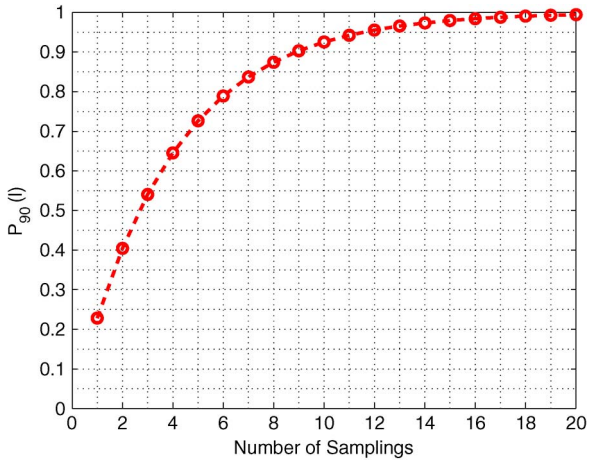


Fig. 12. Probability that Tor bridge users try ℓ times to derive at least one unblocked bridge.

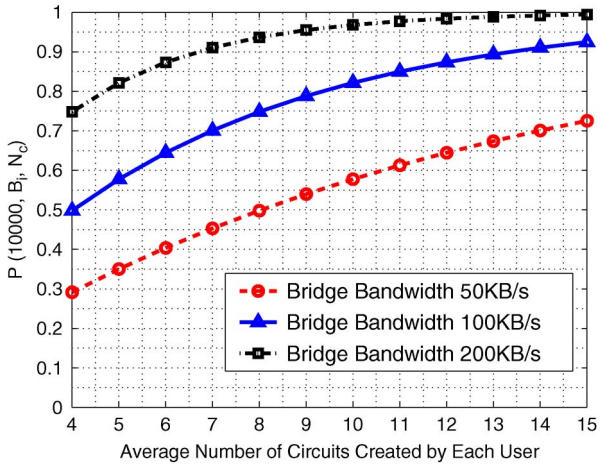


Fig. 14. Probability that at least one circuit traverses both a bridge with bandwidth B_i and our controlled middle router with total bandwidth M'.

the Tor bridges could be discovered by using controlled middle routers. Based on the real-world data from *Tor Metric Portal* [20], there are around 25,000 Tor bridge users daily. Denote N_c as the average number of circuits created by each user. According to our experimental results in Fig. 4, the selection of a bridge is actually based on its bandwidth. Bridges with higher bandwidths will be used by more users and traverse more circuits from bridge users.

Assume that the bandwidth of a bridge i and the total bandwidth of all bridges are B_i' and B', respectively. Then, the number of circuits that traverse the bridge i is $25000 * \frac{B_i'}{B'} * N_c$. If each Tor client establishes N_c circuits, the probability that at least one circuit that traverses both the bridge i and our controlled middle router with the total bandwidth M' can be obtained by $P(M', B_i', N_c) = 1 - (1 - P(M'))^{25000 * \frac{B_i'}{B'} * N_c}$. Then, based on this, we can derive the probability that, after d days, the bridge i can be discovered by our controlled middle router with a total bandwidth of M'. The probability is determined by, $P(d, M', B_i', N_c) = 1 - (1 - P(M', B_i', N_c))^d$.

Assume that we control only one middle router with bandwidth 10 MB/s, i.e., $M = 10000$ KB/s. Fig. 15 illustrates the relationship between the probability $P(10000, B_i, N_c)$, N_c and B_i. From this figure, we can observe that the probability $P(10000, B_i, N_c)$ increases as N_c (the average number of circuits created by each Tor client), and B_i (the bandwidth of the bridge i). Based on our observation at a Tor client, a Tor client creates at least 4 circuit once it is launched. The default advertised bandwidth of a bridge is 100 KB/s. Hence, we can see that when the average number of circuit creation for each Tor client approaches 15, the probability $P(10000, 100, 15)$ exceeds 90 percent. Furthermore, we assume the worst case in which each Tor client only creates 4 circuits. Fig. 15 shows the probability $P(d, 10000, B_i', 4)$ that the bridge i can be discovered by our controlled middle router. As we can see from this figure, the bridges with bandwidth 100 KB/s and 200 KB/s can be 100 percent discovered in 7 days, while the bridge with bandwidth 50 KB/s can be 100 percent discovered in 10 days. These results explain the reason why we could discover so many bridge over two weeks in our real-world experiments. In addition, it demonstrates that the Tor bridge discovery approach through the controlled middle router could be much more effective than the discovery approaches through HTTPS and email.

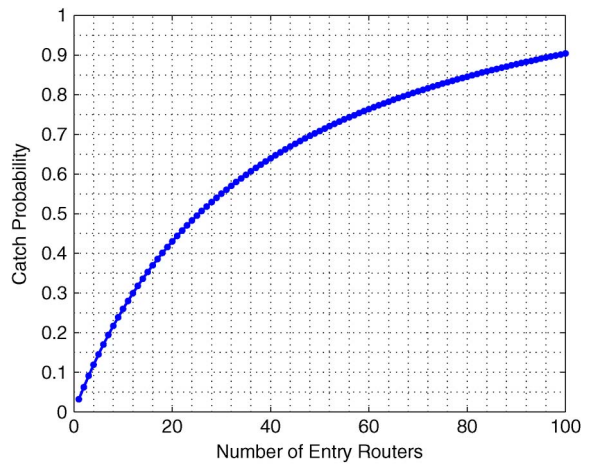


Fig. 13. Probability that a circuit chooses the malicious entry routers vs. number of malicious Tor entry routers.

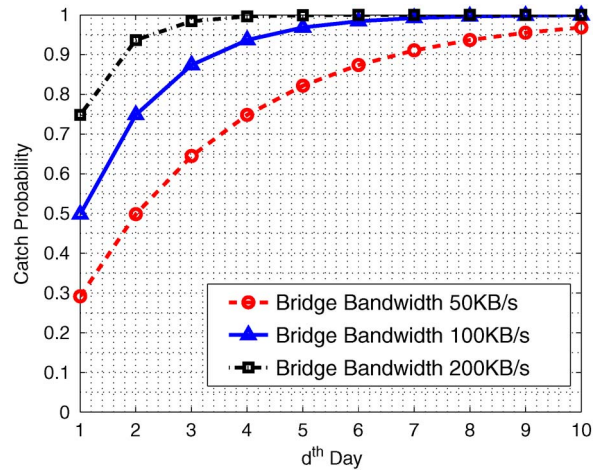


Fig. 15. Probability that the bridge i can be discovered by our controlled middle router after d days.

6 Related Work

In recent years, various anonymous communications systems, including Tor [1], Anonymizer [21], I2P [22], Salsa [23], and Torsk [24] have been designed and deployed to provide online privacy and censorship resistance. There are lots of research efforts on traffic analysis against these systems. For example, Wang *et al.* [25] investigated the feasibility of a timing-based watermarking scheme in identifying the encrypted peer-to-peer VoIP calls. Peng *et al.* [26] analyzed the secrecy of timing-based watermarking traceback proposed in [27], based on the distribution of traffic timing. Yu *et al.* [28] proposed a flow marking scheme based on the direct sequence spread spectrum (DSSS) technique. This approach could be used by the adversary to secretly confirm the communication relationship via mix networks. Ling *et al.* [29] proposed the cell counter based attack against Tor that the adversary embeds a signal into the variation of cell counter of the target traffic by varying the counter of cells in the target traffic at the malicious exit onion router. Houmansadr and Borisov [30] investigated a scalable watermark technique to encode the watermarks by changing the locations of packets within selected time slots.

McLachlan *et al.* [5] investigated the weakness of current bridge architecture, leading to a few advanced attacks on the anonymity of bridge operators. Their results indicate that the existing attacks may expose clients to additional privacy risks and Tor exit routers should be considered as sharing a single IP prefix that is mentioned in the bridge design [19]. Vasserman *et al.* [4] presented the attacks against Tor bridges and discussed countermeasures using DHT-based overlay networks. Bauer *et al.* [31] showed that an adversary who controls only six malicious Tor routers could compromise over 46 percent of all clients' circuits in an experimental Tor network with 66 total routers. Edman *et al.* [32] identified the risk associated with a single autonomous system (AS), which observes both ends of an anonymous Tor connection is greater than previously thought. Their results showed that the growth of the Tor network had only a small impact on the network's robustness against an AS-level adversary. Fu *et al.* [33] investigated a cloud computing based approach that deploys high-bandwidth Amazon EC2 sentinels into the Tor network. Their study showed that with the high bandwidth and appropriate number of sentinels, one can achieve a high probability that a Tor circuit passes through an entry sentinel and an exit sentinel.

Wilde [34] and Winter [35] studied the blocking mechanism used by Great Firewall of China. They found that the Great Firewall of China adopts the deep packet inspect (DPI) to detect the Tor protocol feature and then pretend to be a Tor client to establish a Tor connection to the remote machine. Once the connection is successfully established, the bridge is confirmed and then blocked. To defeat this attack, Tor developed a tool ''obfsproxy'', referred to as pluggable transports [36], for circumvention, to obfuscate the Tor protocol. From the Tor client's point of view, the tool acts as a SOCKS proxy and relays the traffic from Tor clients into the pluggable transport to hide the Tor traffic into other type of traffic. Moghaddam *et al.* [37] obfuscated the Tor traffic between the Tor client and bridge as Skype traffic.

Bridge discovery can facilitate the law enforcement to determine whether a suspicious user is using a bridge to communicate with others through the Tor network. As mentioned in Section 2, a bridge is deployed as a hidden entry onion route in a circuit. Consequently, by observing the outgoing traffic of a user, it is non-trivial to determine whether the user exploits a bridge to connect to the Tor network. In addition, a hidden server can use a bridge to further hide the real location of the hidden server. Based on existing research, Tor hidden servers have been wildly abused for various criminal purposes, including the deployment of Botnet command and control (C&C) server [38], [39] and hosting the black market [40] to sell pornography, narcotics, weapons, and others.

A number of research efforts have been conducted to discover Tor hidden servers [41], [42], [43], [44]. For example, Øverlier and Syverson [41] proposed the packet counting based traffic analysis to identify a hidden server at entry onion routers. Biryukov *et al.* [44] studied how to deploy the hidden service directory to harvest hidden service information and investigated the packet counting based traffic analysis to locate hidden servers. Ling *et al.* [43] investigated a Tor protocol-based approach to discover a hidden server from network forensics aspect. Nevertheless, if the hidden server uses several bridges as entry routers, only the bridges, not the hidden server, can be discovered. The bridge discovery approaches investigated in this paper can facilitate the law enforcement to confirm whether the discovered router is a bridge or not. Then, the law enforcement can apply for a search warranty to request this bridge to collaborate and locate the hidden server.

7 Conclusion

In this paper, we conducted extensive analysis and large-scale empirical evaluation on Tor bridge discovery through email, https and malicious Tor middle routers. To discover bridges automatically, we developed a command-and-control architecture on PlanetLab to send emails through Yahoo SMTP to the bridge email server and download bridge webpages from the bridge web server, respectively. We formalized the email and https bridge discovery process as a weighted coupon collector problem and analyzed the expected number of retrieved bridges with a number of samplings. We also exploited the weighted bandwidth routing algorithms on Tor and studied the bridge discovery through malicious Tor middle onion routers deployed on PlanetLab and in an apartment. We formally analyzed the catch probability of discovering bridges through middle onion routers. **Since the number of Tor bridges is not publicly available, we cannot derive the exact total number of Tor bridges during the experimental period. However, our real-world implementation and large-scale experiments demonstrated the effectiveness and feasibility of the three bridge discovery approaches that we investigated in this paper**. In particular, **we have discovered 2365 Tor bridges through email and https and 2369 bridges by only one controlled Tor middle router in 14 days**. Our study shows that the bridge discovery approach based on malicious middle routers is simple, efficient and effective to discover bridges with little

overhead. The discussion of other issues related to bridge discovery and potential mechanisms to counter bridge discovery can be found in Appendix E of supplemental file available online.

ACKNOWLEDGMENT

This work was supported in part by National Key Basic Research program of China under grant 2010CB328104, National Natural Science Foundation of China under Grants 61272054, 61202449, and 61003257, by U.S. NSF Grants 1116644, 0942113, 0958477, 0943479, and 1117175, by China Specialized Research Fund for the Doctoral Program of Higher Education under grant 20110092130002, China National Key Technology R&D Program under Grants No. 2010BAI88B03 and No. 2011BAK21B02, China Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20110092130002, Science Research Foundation of Graduate School of Southeast University, Jiangsu Provincial Key Laboratory of Network and Information Security under grants BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grants 93K-9. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. Z. Ling is the corresponding author.

REFERENCES

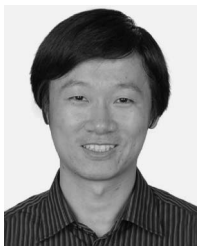
[1] R. Dingledine, N. Mathewson, and P. Syverson, ''Tor: The Second-Generation Onion Router,'' in *Proc. 13th USENIX Secur. Symp.*, Aug. 2004, p. 21.
[2] Tor and Censorship: Lessons Learned, 2010. [Online]. Available: https://blog.torproject.org/blog/tor-and-censorship-lessons-learned
[3] J.B. Kowalski and K. Gabert, *Tor Network Status*, 2010. [Online]. Available: http://torstatus.blutmagie.de/
[4] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim, ''Membership-Concealing Overlay Networks,'' in *Proc. 16th ACM Conf. CCS*, Nov. 2009, pp. 390-399.
[5] J. McLachlan and N. Hopper, ''On the Risks of Serving Whenever You Surf: Vulnerabilities in Tor's Blocking Resistance Design,'' in *Proc. WPES*, Nov. 2009, pp. 31-40.
[6] The Trustees of Princeton University, ''PlanetLab—An Open Platform for Developing, Deploying, Accessing Planetary-Scale Services,'' 2010. [Online]. Available: http://www.planet-lab.org/
[7] R. Dingledine and N. Mathewson, *Tor Directory Protocol, Version 3*, 2010. [Online]. Available: https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/dir-spec.txt
[8] R. Dingledine and N. Mathewson, *Tor Path Specification* 2008. [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt
[9] Imacros 2010. [Online]. Available: http://www.iopus.com/imacros/
[10] Relays in the Tor Network 2010. [Online]. Available: http://metrics.torproject.org/consensus-graphs.html
[11] B.N. Chun, *Pssh*, 2010. [Online]. Available: http://www.theether.org/pssh/
[12] M.A. Muquit, *Mailsend—Send Mail via SMTP Protocol*, 2008. [Online]. Available: http://www.muquit.com/muquit/software/mailsend/mailsend.html
[13] M. Lambers, *Mpop: A POP3 Client*, 2010. [Online]. Available: http://mpop.sourceforge.net/
[14] Freepops, 2010. [Online]. Available: http://www.freepops.org/en/
[15] R. Dingledine and N. Mathewson, *TC: A Tor Control Protocol (Version 1)*, 2010. [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=control-spec.txt
[16] Vidalia, 2010. [Online]. Available: http://www.torproject.org/vidalia/
[17] Amazon.com: Amazon Elastic Compute Cloud (Amazon EC2), 2010. [Online]. Available: http://aws.amazon.com/ec2/
[18] P. Berenbrink and T. Sauerwald, ''The Weighted Coupon Collector's Problem and Applications,'' in *Proc. 15th Annu. Int'l Conf. Comput. Combinatorics*, 2009, pp. 449-458.
[19] R. Dingledine and N. Mathewson, *Design of a Blocking-Resistant Anonymity System*, 2008. [Online]. Available: https://svn.torproject.org/svn/projects/design-paper/blocking.html
[20] The Tor Project, Inc.Tor Metrics Portal 2013. [Online]. Available: https://metrics.torproject.org/
[21] Anonymizer, Inc., 2010. [Online]. Available: http://www.anonymizer.com/
[22] I2P Anonymous Network, 2012. [Online]. Available: http://www.i2p2.de/
[23] A. Nambiar and M. Wright, ''Salsa: A Structured Approach to Large-Scale Anonymity,'' in *Proc. 13th ACM Conf. CCS*, Oct. 2006, pp. 17-26.
[24] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, ''Scalable Onion Routing with Torsk,'' in *Proc. 16th ACM Conf. CCS*, Nov. 2009, pp. 590-599.
[25] X. Wang, S. Chen, and S. Jajodia, ''Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet,'' in *Proc. 12th ACM Conf. CCS*, Nov. 2005, pp. 81-91.
[26] P. Peng, P. Ning, and D.S. Reeves, ''On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques,'' in *Proc. IEEE SP Symp.*, May 2006, pp. 334-349.
[27] X. Wang and D.S. Reeves, ''Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Inter-Packet Delays,'' in *Proc. ACM Conf. CCS*, Nov. 2003, pp. 20-29.
[28] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, ''DSSS-Based Flow Marking Technique for Invisible Traceback,'' in *Proc. IEEE Symp. S&P*, May 2007, pp. 18-32.
[29] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, ''A New Cell Counter Based Attack against Tor,'' in *Proc. 16th ACM Conf. CCS*, Nov. 2009, pp. 578-589.
[30] A. Houmansadr and N. Borisov, ''SWIRL: A Scalable Watermark to Detect Correlated Network Flows 18th Annu. Network Distributed System Security Symp. (NDSS), San Diego, CA, USA, 2011.
[31] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, ''Low-Resource Routing Attacks against Tor,'' in *Proc. WPES*, Washington, DC, USA, Oct. 2007, pp. 11-20.
[32] M. Edman and P.F. Syverson, ''AS-Awareness in Tor Path Selection,'' in *Proc. ACM Conf. CCS*, Nov. 2009, pp. 380-389.
[33] X. Fu, Z. Ling, W. Yu, and J. Luo, ''Cyber Crime Scene(C^2SI) through Cloud Computing,'' in *Proc. 1st Workshop SPCC*, June 2010, pp. 26-31.
[34] T. Wilde, *Great Firewall Tor Probing Circa 09 DEC 2011*, 2011. [Online]. Available: https://gist.github.com/da3c7a9af01d74cd7de7
[35] P. Winter and S. Lindskog, ''How the Great Firewall of China is Blocking Tor,'' in *Proc. 2nd USENIX Workshop FOCI*, 2012, pp. 1-7.
[36] J. Appelbaum and N. Mathewson, *Pluggable Transports for Circumvention*, 2010. [Online]. Available: https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/180-pluggable- transport.txt
[37] H.M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, ''SkypeMorph: Protocol Obfuscation for Tor Bridges,'' in *Proc. 19th ACM Conf. CCS*, 2012, pp. 97-108.
[38] D. Brown, *Resilient Botnet Command and Control with Tor* 2010. [Online]. Available: https://www.defcon.org/images/defcon-18/dc-18-presentations/D.Brown/DEFCON-18-Brown-TorCnC.pdf
[39] Anonymous, ''IAmA a Malware Coder and Botnet Operator,'' AMA 2012. [Online]. Available: http://www.reddit.com/r/IAmA/comments/sq7cy/
[40] N. Christin, ''Traveling the Silk Road: A Measurement Analysis of a Large Anonymous Online Marketplace,'' in *Proc. 22nd Int'l WWW Conf.*, 2013, pp. 213-224.
[41] L. Øverlier and P. Syverson, ''Locating Hidden Servers,'' in *Proc. IEEE S&P Symp.*, 2006, p. 114.
[42] S.J. Murdoch, ''Hot or Not: Revealing Hidden Services by Their Clock Skew,'' in *Proc. 13th ACM Conf. CCS*, Nov. 2006, pp. 27-36.
[43] Z. Ling, J. Luo, K. Wu, and X. Fu, ''Protocol-Level Hidden Server Discovery,'' in *Proc. 32th IEEE Int'l Conf. INFOCOM*, 2013, pp. 1043-1051.
[44] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, ''Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization,'' in *Proc. 34th IEEE Symp. S&P*, 2013, pp. 80-94.
[45] S.M. Bellovin, ''A Technique for Counting NATted Hosts,'' in *Proc. 2nd IMW*, 2002, pp. 267-272.
[46] S. Le Blond, C. Zhang, A. Legout, K. Ross, and W. Dabbous, ''I Know Where You Are and What You Are Sharing: Exploiting P2P Communications to Invade Users Privacy,'' in *Proc. 11th IMC*, 2011, pp. 45-60.

[47] A. Castiglione, A.D. Santis, U. Fiore, and F. Palmieri, ''Device Tracking in Private Networks via NAPT Log Analysis,'' in *Proc. 6th Int'l Conf. IMIS*, 2012, pp. 603-608.
[48] T. Kohno, A. Broido, and K.C. Claffy, ''Remote Physical Device Fingerprinting,'' in *Proc. 26th IEEE Symp. S&P*, 2005, pp. 211-225.
[49] M. Naor, *Verification of a Human in the Loop or Identification via the Turing Test*, 1997. [Online]. Available: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps
[50] CAPTCHA King, 2010. [Online]. Available: http://www.captchaking.com/
[51] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G.M. Voelker, and S. Savage, ''Re: CAPTCHAs-Understanding CAPTCHA-Solving Services in an Economic Context,'' in *Proc. 19th USENIX Security Symp.*, 2010, p. 28.
[52] E. Bursztein, M. Martin, and J. Mitchell, ''Text-Based CAPTCHA Strengths and Weaknesses,'' in *Proc. 18th ACM Conf. CCS*, 2011, pp. 125-138.
[53] J. Isacenkova and D. Balzarotti, ''Measurement and Evaluation of a Real World Deployment of a Challenge-Response Spam Filter,'' in *Proc. 11th IMC*, 2011, pp. 413-426.
[54] Spamhelp, 2012. [Online]. Available: http://www.spamhelp.org/services/listings/challenge-response/
[55] R. Pries, W. Yu, S. Graham, and X. Fu, ''On Performance Bottleneck of Anonymous Communication Networks,'' in *Proc. 22nd IEEE IPDPS*, Apr. 14-28, 2008, pp. 1-11.
[56] Research Problems: Ten Ways to Discover Tor Bridges, 2011. [Online]. Available: https://blog.torproject.org/blog/research-problems-ten-ways-discover-tor-bridges
[57] Freenet, 2012. [Online]. Available: https://freenetproject.org/
[58] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, ''Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity,'' in *Proc. 14th ACM Conf. CCS*, Oct. 2007, pp. 92-102.
[59] P. Mittal and N. Borisov, ''Information Leaks in Structured Peer-to-Peer Anonymous Communication Systems,'' in *Proc. 15th ACM Conf. CCS*, Oct. 2008, pp. 267-278.
[60] A. Tran, N. Hopper, and Y. Kim, ''Hashing it Out in Public: Common Failure Modes of DHT-Based Anonymity Schemes,'' in *Proc. WPES*, Nov. 2009, pp. 71-80.
[61] A. Johnson, P. Syverson, R. Dingledine, and N. Mathewson, ''Trust-Based Anonymous Communication: Adversary Models and Routing Algorithms,'' in *Proc. 18th ACM Conf. CCS*, 2011, pp. 175-186.
[62] P. Mittal, M. Wright, and N. Borisov, ''Pisces: Anonymous Communication Using Social Networks,'' in *Proc. 20th Annu. NDSS Symp.*, 2013, pp. 1-18.

Zhen Ling is a lecturer at the School of Computer Science and Engineering at the Southeast University, Nanjing, China. He received the BS degree (2005) and PhD degree (2014) in computer science from Nanjing Institute of Technology, China and Southeast University, China, respectively. He joined Department of Computer Science at the City University of Hong Kong from 2008 to 2009 as a research associate and then joined Department of Computer Science at the University of Victoria in 2011 as a visiting scholar. His research interests include network security, privacy, and forensics.

Junzhou Luo received the BS degree in applied mathematics and the MS and PhD degrees in computer network both from Southeast University, Nanjing, China, in 1982, 1992, and 2000, respectively. He is a Full Professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests are next generation network, protocol engineering, network security and management, grid and cloud computing, and wireless LAN. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design.

Wei Yu received the BS degree in electrical engineering from Nanjing University of Technology, China, in 1992, the MS degree in electrical engineering from Tongji University, Shanghai, China, in 1995, and the PhD degree in computer engineering from Texas A&M University, College Station, in 2008. He is currently an associate professor with the Department of Computer and Information Sciences, Towson University, Maryland. He was with Cisco Systems Inc. for nine years. His research interests include cyberspace security, computer networks, and cyber-physical systems. He is a recipient of a 2014 NSF CAREER Award and a 2015 University System of Maryland (USM) Regents Faculty Award for Excellence in Scholarship, Research, or Creative Activity.



Ming Yang received the MSc and PhD degrees in computer science and engineering, in 2002 and 2007, respectively, from Southeast University, Nanjing, China. He is currently an Associate Professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests include network security and privacy.

Xinwen Fu received the BS and MS degrees in electrical engineering from Xi'an Jiaotong University, China, in 1995 and University of Science and Technology of China, China, in 1998. He received the PhD degree in computer engineering from Texas A&M University, in 2005. He is an Associate Professor in the Department of Computer Science, University of Massachusetts Lowell. From 2005 to 2008, he was an Assistant Professor with the College of Business and Information Systems at Dakota State University. In summer 2008, he joined University of Massachusetts Lowell as a faculty member. His current research interests are in network security and privacy.