Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery

Zhen Ling*, Junzhou Luo*, Wei Yu‡, Ming Yang* and Xinwen Fu†
*Southeast University, Email: {zhenling, jluo, yangming2002}@seu.edu.cn
‡Towson University, Email: wyu@towson.edu
†University of Massachusetts Lowell, Email: xinwenfu@cs.uml.edu

***Abstract*—Tor is a well-known low-latency anonymous communication system that is able to bypass 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 analyzed the effectiveness of two categories of bridge-discovery approaches: (i) enumerating bridges from bridge https and email servers, and (ii) inferring bridges by malicious Tor middle routers. Large-scale 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 the mechanisms to counter the malicious bridge discovery.**

***Keywords*-Anonymous Communication, Tor, Bridge Discovery, Cloud Computing**

I. 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 various censorship [2]. Because Tor uses source routing for 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*. A bridge can act as the first hop relaying user traffic into the core Tor network. The bridge information 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 can 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 also distributed through various social networks.

Nevertheless, our study and other related work [4], [5] have shown two categories of bridge-discovery approaches: (i) the enumeration of bridges via bulk emails and Tor's https server, and (ii) the use of malicious middle routers to discover bridges (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, censorship wins the battle against 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 censorship in this paper. We conducted an extensive theoretical analysis on two bridge-discovery approaches and our experimental results demonstrate the effectiveness of large-scale bridge discovery in real-world environments. Although there is one related work on discovering bridges [4], [5], the discussion 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.

We formalize the bridge discovery via email and https enumeration as a weighted coupon collector problem and derive the expected number of bridges in terms of the 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], via which emails are sent from 2000 yahoo email accounts in a round robin fashion. We also use the 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 generated 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 how many fresh bridges can be obtained by an IP or a subnet in a time period. Hence, the two enumeration approaches are not efficient and effective to some extent.

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 malicious middle router, the middle router will find that the bridge IP is not within the public Tor router list and thus determine the bridge. Based on our real-world experiments, we are able to prove that with three malicious middle routers of 10MB/s, if 30 circuits are established through a bridge, the probability of discovering the bridge approaches 100%. In other words, if 30 clients use the same bridge to create a circuit, that bridge will be exposed in the probability of almost 100%. Our real-world experimental data show that the 21^{th} circuit created by a bridge client traverses one of 500 PlanetLab nodes of 50KB/s. Our analytical results in Theorem 3 shows that the effectiveness of bridge discovery is determined by the total bandwidth contributed by those malicious middle routers. Using one malicious middle router of 10MB/s, we discovered 2369 Tor bridges over two weeks and validate 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.*

The rest of the paper is organized as follows. In Section II, we introduce the components of Tor and bridges along with the basic operation of Tor for both normal clients and bridge clients. In Section III, we present our approaches for discovering Tor bridges via email, https, and Tor middle routers. In Section IV, we analyze the effectiveness of those bridge discovery approaches. In Section V, we show experimental results on Tor and validate our theory. We present a set of guidelines to counteract those discovery approaches in Section VI. We review related work in Section VII and conclude the paper in Section VIII.

II. 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' viewpoints. Notice that Tor algorithms presented in this paper were discovered from reading the Tor code and include some details, which are not provided by Tor documents. Such details make our analysis complete and match the real Tor behavior.

A. Basic Components of Tor and Bridge

Figure 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: *normal clients* use the Tor core network directly and *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 bridge information by accessing the bridge https/email server or get it from social networks directly.

B. Normal Clients Using Tor

In order 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 path length. Algorithm 1 presents the path selection algorithm. Tor does not choose the same router twice for the same path [8]. From Algorithm 1, we know that in order 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: pure entry router (entry guard), pure exit router, both entry and exit router (denoted as *EE* router), and 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 $max(median,\ 250KB/s)$ [7]. A router is called 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 performance, Tor adopts weighted bandwidth routing algorithms. We use default path length of 3 in Figure 1 to illustrate how a path is selected for 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. Algorithm 2 presents the node exclusion algorithm for

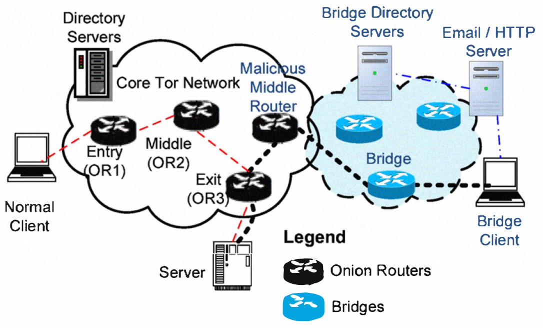

Fig. 1. Tor Network

Algorithm 1 Tor Path Selection Algorithm at Client

```
1: Create a new circuit and initialize global circuit ID, etc.
2: Add the circuit into the global circuit list
3: Decide a suitable length (3 by default) for the circuit
4: if Option 'ExitNodes' is defined in the configuration file then
5:     Use one of 'ExitNodes' as the exit router
6: else
7:     Exclude nodes by Algorithm 2
8:     Select an exit node by Algorithm 3
9: end if
10: Exclude the chosen exit node and select an entry node by Algorithm 4
11: Exclude the chosen exit node and entry node, and select a middle node by Algorithm 4
```

Algorithm 2 Node Exclusion for Selecting Exit Nodes

```
1: Exclude clients that are possible exit nodes (in case that client is an OR node)
2: Exclude nodes not running or remarked as bad exits
3: Exclude nodes not meeting capacity or uptime requirements
4: Exclude nodes remarked as invalid
5: Exclude exit nodes whose policies reject all traffic
```

Algorithm 3 Bandwidth Weighted Node Selection Algorithm

```
Require:
    (a) $B$, the total bandwidth of the nodes in the node list (b)
    $B_E$, the total bandwidth of the exit nodes in the node list
    (c) $B_G$, the total bandwidth of the guard nodes in the node
    list (d) $q$, the number of the nodes in the list (e) $b[i]$, the
    bandwidth of the $i^{th}$ node in the list (f) $W_E$, the weight
    of the exit nodes (g) $W_G$, the weight of the guard nodes
    (h) $bw$, the weighted bandwidth of the nodes (i) $total_{bw}$,
    the totally weighted bandwidth of the nodes (j) $rand_{bw}$,
    the random sampling bandwidth value from the $total_{bw}$
Ensure: Find a suitable node from the node list
 1: Derive a list of qualified running nodes
 2: Count $B$, $B_E$ and $B_G$
 3: if try to find a exit node then
 4:     $W_E = 1$
 5: else
 6:     $W_E = 1 - B/(3 \times B_E)$
 7: end if
 8: if try to find a guard node then
 9:     $W_G = 1$
10: else
11:     $W_G = 1 - B/(3 \times B_G)$
12: end if
13: if $W_E < 0$ then
14:     $W_E = 0$
15: end if
16: if $W_G < 0$ then
17:     $W_G = 0$
18: end if
19: for $i = 1 : q$ do
20:     if the node is both exit and guard node then
21:         $bw = b[i] \times W_G \times W_E$
22:     else if the node is entry then
23:         $bw = b[i] \times W_G$
24:     else if the node is exit then
25:         $bw = b[i] \times W_E$
26:     else
27:         $bw = b[i]$
28:     end if
29:     $total_{bw} = total_{bw} + bw$
30: end for
31: Randomly sample a bandwidth $rand_{bw}$ from $total_{bw}$
32: for $j = 1 : q$ do
33:     if the node is both exit and guard node then
34:         $temp = temp + b[i] \times W_G \times W_E$
35:     else if the node is entry then
36:         $temp = temp + b[i] \times W_G$
37:     else if the node is exit then
38:         $temp = temp + b[i] \times W_E$
39:     else
40:         $temp = temp + b[i]$
41:     end if
42:     if $temp > rand_{bw}$ then
43:         return the $i^{th}$ node
44:     end if
45: end for
```

Algorithm 4 Selection of an Entry/Middle Node

```
1: Derive a list of qualified running nodes
2: if Bandwidth or a guard node is required then
3:     Use a bandwidth weighted Algorithm 3 to choose one;
       (this is the default branch)
4: else
5:     Choose the middle node randomly
6: end if
```

selecting exit nodes. 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 i^{th} 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 i^{th} exit router. Note that $B_E = B_{exit} + B_{EE}$. Algorithm 3 is used to choose 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 i^{th} entry router from the entry set is $\frac{Bi_G}{B_{entry}+B_{EE}\cdot W_E}$, where

$$W_E = 1 - \frac{B}{3B_E}. \tag{2}$$

W_E is the exit bandwidth weight and Bi_G is i^{th} 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 describes the selection of middle nodes for a circuit[1]. Notice that 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 i^{th} router from the remaining set of Tor routers becomes $\frac{Bi}{B_{exit}\cdot W_E+B_{entry}\cdot W_E+B_{EE}\cdot W_E\cdot W_G+B_{N-EE}}$, where Bi is i^{th} bandwidth in the remaining set and B_{N-EE} is the total bandwidth of N-EE routers.

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

[1]Another constraint is that each router on a path must be selected from a distinct /16 subnet.

path. Once the circuit is established, the client can connect to a web server through the circuit.

C. Bridge Clients Using Tor

In order to access Tor, a bridge client needs to obtain a least one bridge. As we can see from Figure 1, the bridge client can obtain the information of bridges via email and https. We will further discuss these methods in Section III. 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 Figure 1), respectively. Finally, the bridge client creates a circuit and anonymously surf the Internet.

III. 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.

A. Basic Ideas

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

(i) Email and https enumeration. An attacker can use a Yahoo 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.

(ii) 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 an apartment, PlanetLab or Amazon EC2, we may discover bridges from any bridge pool, including those privately distributed in social networks.

B. 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 via 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. Since Yahoo does not provide free POP3 (Post Office Protocol 3) service, we use a tiny POP3 client *Mpop* [13] to retrieve Yahoo emails via 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 bridge IPs embedded in emails.

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

C. Discovering Bridges via HTTPS

Since 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. We introduce the following two schemes to this end:

(i) 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.

(ii) 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 via distinct exit nodes. Once the circuit is created, the tool *wget* is used to download bridge web pages.

D. Discovering Bridges Via Tor Middle Routers

Figure 1 illustrates the basic idea of discovering Tor bridges via middle Tor routers. We deploy malicious Tor middle routers on PlanetLab to discover bridges connected to these Tor middle routers. Recall 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 the 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 bridge IPs to us. This approach allows us to automatically retrieve bridges via 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.

Notice 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 becoming entry routers, we need to reduce their bandwidth or control their uptime. By configuring the exit policy, we also prevent those malicious routers from becoming exit routers.

IV. Analysis

In this section, we first analyze the effectiveness of the bridge discovery via emails and https. 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 via malicious Tor middle routers.

A. Bridge Discovery via Emails and HTTPS

The approaches that enumerate bridges via emails and https can be formalize as a weighted coupon collector problem. In the classical coupon collector problem [18], all n coupons are obtained with an equal probability of $\frac{1}{n}$. To derive the entire collection, the collector needs to collect $\Theta(n \ln n)$ coupons on average. However, bridges are not distributed with an equal probability, but weighted on their bandwidth. This is validated by our real-world experiments in Section V. We now explain the weighted coupon collector problem in our context.

To enhance Tor performance, Tor adopts the weighted bandwidth routing algorithm for path (circuit) selection. Specifically, the higher a router's advertised bandwidth, the higher the chance that the router is selected for a circuit. Recall a bridge can act as a Tor entry router. With this weighted bandwidth routing algorithm, we have the following theorem:

Theorem 1. *In a Tor network with n Tor bridges, we assume that the bandwidth of the bridges comprises a set $\{B_1, B_2, \cdots, B_n\}$. The probability p_i that the i^{th} router of bandwidth B_i can be selected is $p_i = \frac{B_i}{\sum_{i=1}^{n} B_i}$. Denote X_h as the number of distinct bridges after h (≥ 1) circuits are created by the attacker. The expected number of different bridges generated by these h samplings can be computed by*

$$E(X_h) = n - \sum_{i=1}^{n}(1 - p_i)^h. \quad (3)$$

The proof of Theorem 1 is given in Appendix A of our technical report [19]. It can be observed from Theorem 1 that when we sample more (h increases), we can obtain more bridges. Notice that to avoid malicious enumeration, the bridge authority divides the available bridges into pools. Each pool is available in a certain time window [20]. 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 Figure 4.

B. Bridge Discovery via 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 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 $\{B_1, B_2, \cdots, B_k, B_{k+1}, \cdots, B_{k+N}\}$, where $\{B_1, \cdots, B_k\}$ 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 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: *pure entry router* (entry guard), *pure exit router*, *both entry and exit* router (denoted as *EE* router), and *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. Notice that $B = B_{entry} + B_{N-EE} + B_{EE} + B_{exit}$. Based on

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

the weighted bandwidth routing algorithm discussed in Section II-B, the bandwidth weight can be derived by,

$$W_E = \begin{cases} 1 - \frac{B+k\cdot b}{3\cdot(B_{exit}+B_{EE})} & : \ W_E > 0, \\ 0 & : \ W_E \leqslant 0. \end{cases} \quad (4)$$

$$W_G = \begin{cases} 1 - \frac{B+k\cdot b}{3\cdot(B_{entry}+B_{EE})} & : \ W_G > 0, \\ 0 & : \ W_G \leqslant 0. \end{cases} \quad (5)$$

The weighted bandwidth $B_{exit'}$, $B_{EE'}$, $B_{entry'}$ and $B_{N-EE'}$ can be derived as follows,

$$\begin{aligned} B_{exit'} &= B_{exit} \cdot W_E, & (6)\\ B_{EE'} &= B_{EE} \cdot W_E \cdot W_G, & (7)\\ B_{entry'} &= B_{entry} \cdot W_G, & (8)\\ B_{N-EE'} &= B_{N-EE} + k \cdot b. & (9) \end{aligned}$$

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 II-B (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 for calculating the catch probability.

Theorem 2. *The* **catch probability** *can be derived by*

$$P(k, b) = \frac{k \cdot b}{B_{exit'} + B_{EE'} + B_{entry'} + B_{N-EE'}}, \quad (10)$$

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

Theorem 2 is intuitive based on the bandwidth weighted path selection algorithm. From Theorem 2, we derive the following corollaries.

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

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

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.

$$P(k, l) > P(k, b), \text{ where } l > b. \quad (12)$$

The proof of Corollary 1 and Corollary 2 is given in Appendix B of [19]. 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 is not a surprise.

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 IPs. Theorem 3 answers this question.

Theorem 3. *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* $M \geq M'$,

$$P(M) \geq P(M'). \quad (13)$$

The equality holds when $M = M'$.

The proof of Theorem 3 is given in Appendix C of [19]. Theorem 3 implies that an attacker 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 V-B validate this observation.

In practice, we also want to know the catch probability vs. the number of created circuits. Theorem 4 answers this question.

Theorem 4. *After* q *circuits are created, the catch probability that at least one bridge connects to one of the malicious* k *routers of bandwidth* b *can be derived by*

$$P(k, b, q) = 1 - (1 - P(k, b))^q, \text{ where } q = 1, 2, 3, \ldots. \quad (14)$$

Theorem 4 is intuitive. From Theorem 4, we have Corollary 3.

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

$$P(k, b, h) > P(k, b, q), \text{ where } h > q. \quad (15)$$

The proof of Corollary 3 is given in Appendix D of [19].

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 (14).

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*

$$P(M, q) = 1 - (1 - P(M))^q. \quad (16)$$

Corollary 4 is intuitive given Theorem 4. From Theorems 3 and 4, we also derive Corollary 5. The proof is given in Appendix D of [19].

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

$$P(M, q) > P(M', q), \text{ where } M > M'. \quad (17)$$

In summary, the catch probability increases with two factors: the total bandwidth of malicious Tor middle routers and the number of created circuits. Our experimental data in Section V validate these theoretical results well. Our theoretical findings and experimental data show that the bridge discovery approach via Tor middle routers is effective even if we can only control a small number of Tor middle routers.

V. Evaluation

We have implemented the proposed Tor bridge discovery approaches in Section III. 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 IV well.

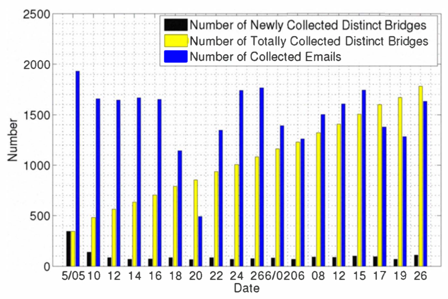

Fig. 2. Discovered Bridges via Emails

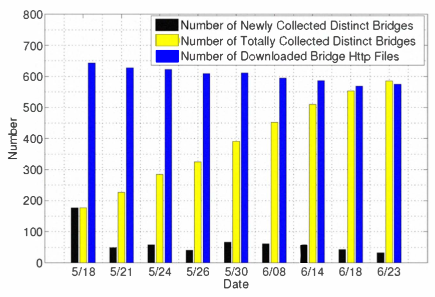

Fig. 3. Discovered Bridges via HTTPS

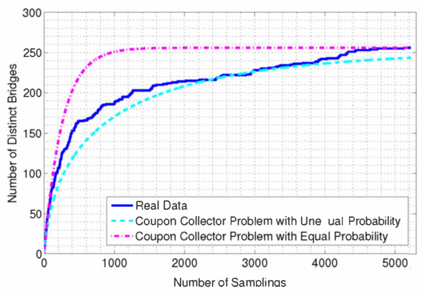

Fig. 4. Number of Samplings v.s. Number of Distinct Bridges

A. Bridge Enumeration via Email and HTTPS

To evaluate bridge-discovery approaches via emails and https, we conducted large-scale experiments from PlanetLab from May to June, 2010. Figure 2 gives the number of newly collected distinct bridges, number of totally collected distinct bridges, and 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 Figure 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.

Figure 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 discovered the new bridges. To enumerate all bridges, we only need to continue experiments. Figure 3 gives the data obtained via https. The number of discovered bridges via https has a similar trend to that in Figure 2.

We now verify Theorem 1 in Section IV using real-world data. Recall that Tor distributes different pools of bridges (there is crossover among pools) via email and https servers over time. However, experiments on a certain 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. Figure 4 shows the relationship between the number of samplings and the number of distinct bridges. It can be observed that the curve of the not-weighted classical coupon collector problem is much steeper than the curve for the real data at the beginning. This implies that the bridges are not distributed uniformly.

In order to verify that the bridge distribution is 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 bridge bandwidth (note that we do not know bridge bandwidth). We then obtain the curve of the weighted coupon collector problem based on Equation (3). Figure 4 shows that the theoretical curve is only slightly lower than the real data generated curve. Hence, it is highly possible that bridges are distributed with their bandwidth as the weight. Such a weighted distribution is also consistent with Tor's weighted routing algorithm for performance enhancement. Actually, Tor developers later confirmed this fact to us.

B. Bridge Discovery via Tor Middle Routers

Figure 5 shows the public Tor router bandwidth cumulative distribution function on July 10, 2010. There were 1604 active Tor routers, including 326 pure entry onion routers, 525 pure exit onion routers, and 132 EE routers. Using the real-world data, Figure 6 shows the relationship between the theoretical catch probability and the number of controlled Tor middle routers based on Theorem 2. As we can see, the catch probability is 14.7% when 512 Tor middle routers with bandwidth 50KB/s are used, i.e., $P(512, 50) = 14.7\%$. Based on Theorem 4, Figure 7 illustrates the catch probability when the bridge clients create q circuits, that is $P(512, 50, q)$. From Figure 7, we can see that in theory, the catch probability approaches 99%, after the bridge clients created 30 circuits, i.e., $P(512, 50, 30) \approx 99\%$. In addition, from Equation (17), 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 50KB/s (because of the limited bandwidth of PlanetLab nodes). To avoid affecting the Tor network, we only conducted a short experiment for 2 days. We set up a Tor client to create 430 circuits via our own Tor bridge in an apartment. We found that the 21^{th} circuit passed through our Tor middle routers in 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. Figure 8 gives the empirical cumulative distribution function (ECDF) of the bandwidth of onion routers 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% of those routers have a bandwidth more than 1MB/s. However, as shown in Figure 5, only 10% of Tor routers have a bandwidth of larger than 1MB/s. This implies that the higher bandwidth the routers have, the higher chance these routers are selected as middle routers for serving circuits.

Figure 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 2. We can see that the theoretical probability is monotonously increasing with both the number of controlled middle routers and their bandwidth advertisement. These observations match our analytical results in Theorems 1 and 2 well. As expected, the catch probability

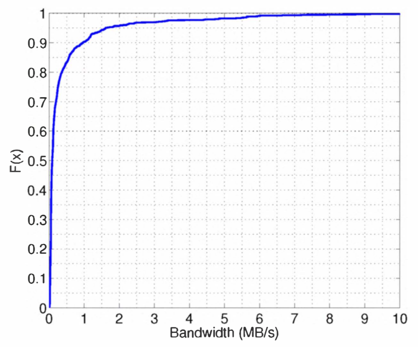

Fig. 5. Empirical CDF of Bandwidth of All Routers in the Tor Network

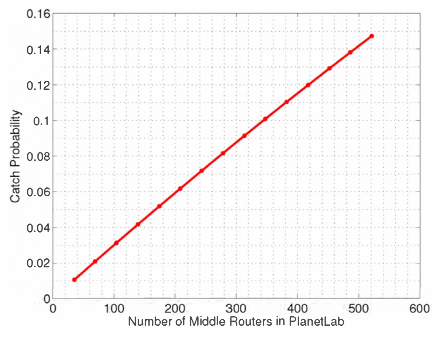

Fig. 6. Probability that a Circuit Chooses the Middle Routers in PlanetLab vs. Number of Tor Middle Routers in PlanetLab

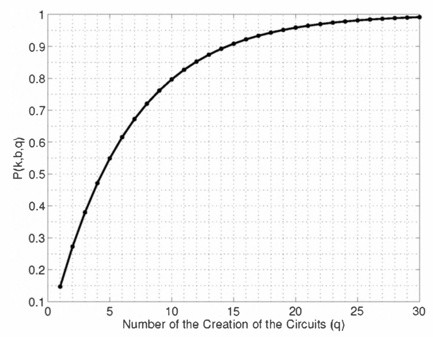

Fig. 7. Probability That at Least a Circuit Traverses Through the Controlled Middle Routers After Bridge Clients Create q Circuits

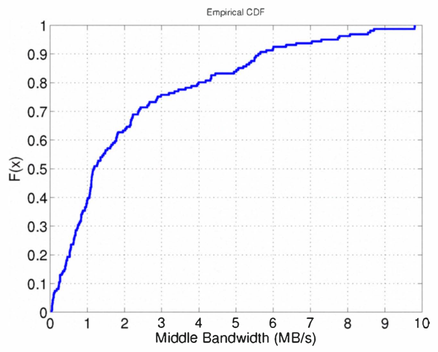

Fig. 8. Empirical CDF of Bandwidth of Selected Middle Routers

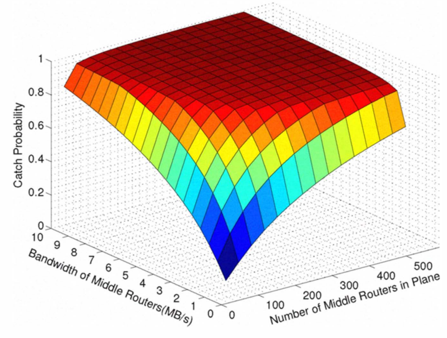

Fig. 9. Probability that a Circuit Chooses the Tor Middle Routers in PlanetLab vs. Number of Tor Middle Routers & Bandwidth Advertisement of Tor Middle Routers in PlanetLab

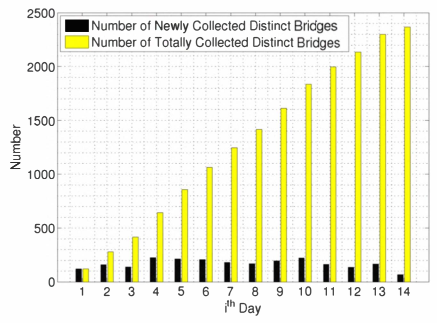

Fig. 10. Discovery Bridges via a Tor Middle Router

approaches 90% when there are 20 malicious middle routers with 10MB/s bandwidth, i.e., $P(20, 10000) \approx 90\%$.

To verify Theorem 3 that the bridge discovery is determined by the aggregated bandwidth of the malicious Tor nodes, we configured a high bandwidth middle router of 10MB/s. We recorded the bridges that pass through this middle router from July 10 to 23, 2010. Figure 10 gives the number of newly discovered distinct bridges and the number of totally collected distinct bridges. The number of totally collected bridges increases over time. Eventually, we obtained 2369 bridges, indicating that discovering bridges via middle routers can be very effective and efficient and the catch probability is mainly determined by total bandwidth contributed by malicious middle routers. Notice that to prevent the middle router from becoming an entry router in 7 days, we restarted the router on the 6^{th} day.

VI. Discussion of Countermeasures

We have demonstrated the impact of our bridge-discovery approaches. We now discuss possible countermeasures against these malicious discovery. We note that those countermeasures can reduce the effectiveness of the malicious bridge discovery, but none of them will be an ultimate solution.

(i) Explore the human interactive proofs (HIPs) method to resist automatically large-scale bridge discovery: CAPTCHA as a known HIP-based approach could be used to increase difficulty to large-scale automatic bridge enumeration through emails and https. A CAPTCHA consists of a visual challenge in the form of alphanumeric characters, which are distorted in such a way that available computer vision algorithms have difficulty segmenting and recognizing the text. The Tor bridge https and email server may adopt CAPTCHA to this end. However, there have been various efforts in automatically deciphering CAPTCHA [21].

(ii) Design new router selection strategies: As we discussed, the weighted bandwidth routing strategy speeds up the bridge discovery through controlled Tor middle onion routers. One naive approach to reduce the impact of malicious Tor middle routers is to revise the routing strategy and select routers for circuits uniformly and randomly. However, this uniform routing strategy will deteriorate Tor network performance because low bandwidth routers will have a high probability of being selected for a circuit.

(iii) Build a backbone of trusted routers: Tor may establish a backbone of trusted routers, to which bridges can connect. This strategy will defeat the bridge discovery based on injecting malicious Tor middle router. However, the centralized backbone may become Tor network performance bottleneck.

(iv) Use Anonymous Peer-to-Peer (P2P) Communication Network: Anonymous P2P systems may provide much robust anti-censorship. We can organize routers into a distributed hash table (DTH). DHT based routing strategies are scalable and flexible for locating routers and creating communication tunnels such as Tor circuits. Because the DHT routing schemes do not depend on a global list of routers, it is hard for a censorship agent to block all the routers quickly. Tor can be transformed into a type of P2P network if most clients

volunteer to act as bridges. Nevertheless, routing schemes based on DHT have security issues as well [22], [23], [24].

VII. Related Work

Because of space limit, we only review the most related work. 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 [20]. Vasserman *et al.* [4] presented the attacks against Tor bridges and discussed countermeasures using DHT based overlay networks. Bauer *et al.* [25] showed that an adversary who controls only 6 malicious Tor routers can compromise over 46% of all clients' circuits in an experimental Tor network with 66 total routers. Edman *et al.* [26] 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.

VIII. Conclusion

In this paper, we conducted extensive analysis and large-scale empirical evaluation on Tor bridge discovery via email, https and malicious Tor middle routers. To discover bridges automatically, we developed a command-and-control architecture on PlanetLab to send emails via 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 Tor weighted bandwidth routing algorithm and studied the bridge discovery via malicious Tor middle onion routers deployed on PlanetLab and in an apartment. We formally analyzed the catch probability of discovering bridges via middle onion routers. Our real-world implementation and large-scale experiments validated the effectiveness and feasibility of the three bridge discovery approaches. **We have discovered 2365 Tor bridges via 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. We also discussed potential mechanisms to counter bridge discovery.

Acknowledgement

This work was supported in part by National Key Basic Research Program of China under Grants 2010CB328104, National Natural Science Foundation of China under Grants 60903162, 60903161, 61070158, 61070161, 61003257, China Specialized Research Fund for the Doctoral Program of Higher Education under Grants 200802860031, Jiangsu Provincial Natural Science Foundation of China under Grants BK2008030, 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, and USA NSF grants 0942113, 0958477, 0943479 and CNS-1117175 and the Army Research Laboratory under grant W911NF-11-1-0193. 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.

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] "Tor and censorship: lessons learned," https://blog.torproject.org/blog/tor-and-censorship-lessons-learned, 2010.

[3] J. B. Kowalski and K. Gabert, "Tor network status," http://torstatus.blutmagie.de/, 2010.

[4] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim, "Membership-concealing overlay networks," in *Proceedings of the 16th ACM conference on Computer and communications security (CCS)*, November 2009.

[5] J. McLachlan and N. Hopper, "On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, November 2009.

[6] The Trustees of Princeton University, "Planetlab — an open platform for developing, deploying, and accessing planetary-scale services," http://www.planet-lab.org/, 2010.

[7] R. Dingledine and N. Mathewson, "Tor directory protocol, version 3," http://tor.eff.org/svn/trunk/doc/spec/dir-spec.txt, 2010.

[8] ——, "Tor path specification," http://tor.eff.org/svn/trunk/doc/spec/path-spec.txt, 2008.

[9] "Imacros," http://www.iopus.com/imacros/, 2010.

[10] "Relays in the tor network," http://metrics.torproject.org/consensus-graphs.html, 2010.

[11] B. N. Chun, "Pssh," http://www.theether.org/pssh/, 2010.

[12] M. A. Muquit, "Mailsend - send mail via smtp protocol," http://www.muquit.com/muquit/software/mailsend/mailsend.html, 2008.

[13] M. Lambers, "Mpop: A pop3 client," http://mpop.sourceforge.net/, 2010.

[14] "Freepops," http://www.freepops.org/en/, 2010.

[15] R. Dingledine and N. Mathewson, "Tc: A tor control protocol (version 1)," https://svn.torproject.org/svn/tor/trunk/doc/spec/control-spec.txt, 2010.

[16] "Vidalia," http://www.torproject.org/vidalia/, 2010.

[17] "Amazon.com: Amazon elastic compute cloud (Amazon EC2)," http://aws.amazon.com/ec2/, 2010.

[18] P. Berenbrink and T. Sauerwald, "The weighted coupon collector's problem and applications," in *Proceedings of the 15th Annual International Conference on Computing and Combinatorics*, 2009.

[19] Z. Ling, X. Fu, W. Yu, J. Luo, and M. Yang, "Extensive analysis and large-scale empirical evaluation of tor bridge discovery," http://www.cs.uml.edu/~xinwenfu/paper/Bridge.pdf, University of Massachusetts Lowell, Tech. Rep., 2011.

[20] R. Dingledine and N. Mathewson, "Design of a blocking-resistant anonymity system," https://svn.torproject.org/svn/projects/design-paper/blocking.html, 2008.

[21] "Captcha king," http://www.captchaking.com/, 2010.

[22] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security? how attacks on reliability can compromise anonymity," in *Proceedings of the 14th ACM Conference on Computer and Communications Security(CCS)*, October 2007.

[23] P. Mittal and N. Borisov, "Information leaks in structured peer-to-peer anonymous communication systems," in *Proceedings of the 15th ACM Conference on Computer and Communications Security(CCS)*, October 2008.

[24] A. Tran, N. Hopper, and Y. Kim, "Hashing it out in public: Common failure modes of dht-based anonymity schemes," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, November 2009.

[25] 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)*, Washington, DC, USA, October 2007.

[26] M. Edman and P. F. Syverson, "As-awareness in tor path selection," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS)*, November 2009.