Low-Resource Routing Attacks Against Tor

Kevin Bauer[†] Damon McCoy[†] Dirk Grunwald[†] Tadayoshi Kohno[‡] Douglas Sicker[†]

[†] Department of Computer Science
University of Colorado
Boulder, CO 80309-0430 USA
{bauerk, mccoyd, grunwald, sicker}@colorado.edu

[‡] Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2969 USA
yoshi@cs.washington.edu

ABSTRACT

Tor has become one of the most popular overlay networks for anonymizing TCP traffic. Its popularity is due in part to its perceived strong anonymity properties and its relatively low latency service. Low latency is achieved through Tor's ability to balance the traffic load by optimizing Tor router selection to probabilistically favor routers with high-bandwidth capabilities.

We investigate how Tor's routing optimizations impact its ability to provide strong anonymity. Through experiments conducted on PlanetLab, we show the extent to which routing performance optimizations have left the system vulnerable to end-to-end traffic analysis attacks from non-global adversaries with minimal resources. Further, we demonstrate that entry guards, added to mitigate path disruption attacks, are themselves vulnerable to attack. Finally, we explore solutions to improve Tor's current routing algorithms and propose alternative routing strategies that prevent some of the routing attacks used in our experiments.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.5 [**Computer - Communication Networks**]: Local and Wide-Area Networks

General Terms

Security, Reliability, Experimentation

Keywords

Anonymity, Tor, Traffic Analysis, Load Balancing

1. INTRODUCTION

We present new methods for compromising the security of the Tor anonymous overlay network [7]. This work focuses upon the following two scientific questions: (1) how can we *minimize* the requirements necessary for any adversary to compromise the anonymity of a flow; and (2) how can we harden Tor against our attacks?

Central to our attacks is the fact that a *lying* adversary — by exaggerating its resource claims — can compromise an unfair percentage of Tor entry and exit nodes. Further, we show how an adversary can compromise the anonymity of a Tor path *before* any data is transmitted, which enables us to further reduce the resource requirements on the attacker. We experimentally evaluate the efficacy of our attacks via experiments with an isolated Tor deployment on PlanetLab. We also explore methods for mitigating the severity of our attacks.

Historical Balance Between Anonymity and Performance. Conventional wisdom suggests that it is impossible for practical privacy-enhancing systems to provide perfect anonymity. Therefore, the designers of such systems must consider restricted threat models. Consider, for example, an anonymous communications system that routes traffic through multiple intermediate nodes. While it is generally possible to perform a traffic analysis attack against a connection if both of the endpoints are compromised, theoretical analyses of anonymous networks show that the likelihood of successfully launching such a traffic analysis attack becomes negligible as the network size increases [7, 23, 26].

When the Tor network was launched, it consisted of few routers transporting little traffic. Consequently, in its initial design, Tor provided no traffic load balancing capability, and it was with respect to this initial design that the above-mentioned theoretical analyses were performed [7]. As the network grew to include nodes with a wide variety of bandwidth capabilities, it became necessary to ensure that the traffic is efficiently balanced over the available resources in order to achieve low latency service. Tor's routing mechanism was modified to prefer high-bandwidth, high-uptime routers that have the resources to accept new connections and transport traffic. Dingledine, Mathewson, and Syverson suggested that a non-uniform router selection mechanism may increase an attacker's ability to compromise the system's anonymity [8], though the full security implications of this load balancing was left to further research.

Our Approach. Within Tor's routing model, an adversary could deploy a few nodes that have — or *appear* to have — high-bandwidth connections and high-uptimes. In the latter case, the adversary is said to *lie* about its resources. With high probability, such an adversary would be able to successfully compromise the two endpoints — the *entry node* and the *exit node* — of a new Tor client's connections. Compromising the entry and exit nodes with non-uniform probability is the first step in our attack.

WPES'07, October 29, 2007, Alexandria, Virginia, USA.

As noted above, previous works showed that, upon compromising the entry and exit nodes, it is possible to compromise the anonymity of a connection via traffic analysis. However, in the spirit of *minimizing* the resource requirements for the adversary, we develope an end-to-end method for associating a client's request to its corresponding destination *before* any payload data is sent. This is important since low-resource malicious nodes may lack the bandwidth to forward significantly many data packets.

Experimental Verification. We experimentally show, using an isolated Tor deployment on PlanetLab, that adversaries with sparse resources — such as adversaries with a few nodes behind residential cable modems — can compromise the anonymity of many paths for new clients. In a Tor deployment of 60 honest and 6 malicious Tor routers, our attack compromised over 46% of the path-building requests from new clients through the network. This illustrates the inherent difficulty of simultaneously attempting to provide optimal anonymity and efficient use of the network's resources.

Prior Attacks. Other attacks against Tor have focused upon traffic analysis and locating hidden services. Murdoch and Danezis presented a low cost traffic analysis technique that allowed an outside observer to infer which nodes are being used to relay a circuit's traffic [17], but could not trace the connection to the initiating client. Øverlier and Syverson demonstrated a technique for locating hidden services that used false resource claims to attract traffic [19]. Murdoch and Zieliński [18] analyze the route selection problem in Tor in the context of Internet exchanges (IXes), and give a new traffic analysis technique for processing data from IXes. We extend the attack on hidden services to effectively compromise the anonymity of general-purpose paths using resource-constrained nodes.

Attack Variants and Improvements. We consider additional attack variants and improvements in the body of this paper. For example, we show how to adapt our attack to compromise the flows of pre-existing Tor clients; recall that our attack as described above is (generally) only successful at compromising new clients, who have not already picked their preferred entry nodes. We also consider further resource reductions, such as using watermarking techniques to, in some cases, eliminate the need for a compromised exit node. An important extension is an attack upon the entry guard selection process, where we show that it is possible to displace all legitimate entry guards with malicious nodes. Additionally, we consider methods to improve the effectiveness of our attack, such as a variant of the Sybil attack [10].

Countermeasures. Next we explore counter-measures to routing attacks in Tor. High-resource adversaries, even if only in possession of a few malicious nodes, seem to pose a fundamental security challenge to any high-performance, multi-hop privacy enhancing system. We focus on designing solutions to mitigate the low-resource attacker's ability to compromise anonymity. These solutions include verifying information used in routing decisions, allowing clients to make routing decisions based on observed performance, and implementing location diversity in routers to mitigate Sybil attacks.

Context. Following the initial disclosure of the primary vulnerability behind our attacks [2], development began by the Tor community on measures to mitigate the effectiveness of these attacks. While an adversary's ability to launch a large number of malicious nodes from the same physical machine or network has been partially addressed [1], the more challenging problem of verifying bandwidth claims remains open.

Outline. The remainder of this paper is organized as follows: In Section 2, we describe the Tor system architecture and its routing algorithms. Section 3 explains the attack and the path linking algorithm and in Section 4, we present the experimental setup and results. Additional attack extensions are considered in Section 5 and proposed defenses are discussed in Section 6. Finally, we provide concluding remarks in Section 7.

2. BACKGROUND

In order to present the methodology used in our experiments, we first provide a brief overview of the Tor system architecture, an in depth analysis of Tor's router selection algorithms, and a description of Tor's attack model.

2.1 Understanding Tor at a High Level

The Tor project's main goal is to develop a network that protects the privacy of TCP connections. In addition, Tor aims to provide end-user anonymity with constraints such as low-latency, deployability, usability, flexibility, and simple design. Currently, Tor can anonymize TCP streams, providing a relatively high-throughput and low-latency onion routing network [14].

In the Tor architecture, there are several fundamental concepts which are defined as follows: A *Tor router* is the server component of the network that is responsible for forwarding traffic within the core of the network. A *Tor proxy* is the client part of the network that injects the user's traffic into the network of Tor routers; for our purposes, one can view the Tor proxy as a service that runs on the user's computer. A *circuit* is a path of three routers (by default) through the Tor network from the proxy to the desired destination server. The first router on the circuit is referred to as the *entry* router, the second router is called a *middle* router, and the final hop is the *exit* router. Tor proxies choose stable and high bandwidth routers to be *entry guards*, which are used as an entry router. We use the terms *entry guard* and *entry router* synonymously throughout this paper. Router information is distributed by a set of well-known and trusted *directory servers*. Finally, the unit of transmission through the network is called a *cell*.

At the core of Tor is a circuit switched network. The circuits are carefully built in such a way that it is intended to be complex and resource intensive for eavesdropping attackers and malicious nodes within the network to link the originator of a circuit to the destination. Cells are encrypted by the originator of the circuit using a layered encryption scheme. Each hop along the circuit removes a layer of encryption until the cell reaches the exit node at the end of the circuit and is fully decrypted, reassembled into a TCP packet, and forwarded to its final destination. This process is known as *onion routing* [14]. For a thorough evaluation of the security of Tor's circuit building algorithm, we refer the reader to Goldberg [13] and more details about the cryptography used in Tor can be found in its design document [7].

Tor can operate as both a proxy, which builds circuits to forward a local user's traffic through the network and also as a router, which will accept connections from other routers/proxies and forward their traffic as well as the local

user's traffic. By default, Tor currently operates as a proxy (client), handling only the local user's traffic.

2.2 Tor's Router Selection Algorithms

There are currently (as of `Tor` version 0.1.1.23) two parts to the algorithm that Tor uses to select which routers to include in a circuit. The first part is used to select the entry router, and the second part is used to select subsequent routers in the circuit. We will show methods to exploit both of these algorithms, as currently implemented in Tor, in Section 3.

Entry Router Selection Algorithm. The default algorithm used to select entry routers was modified in May 2006 with the release of Tor version 0.1.1.20. Entry guards were introduced to protect circuits from selective disruption attacks, thereby reducing the likelihood of an attacker intentionally breaking circuits until they are on a target victim's circuit [19]. The entry guard selection algorithm works by automatically selecting a set of Tor routers that are marked by the trusted directory servers as being "fast" and "stable." The directory server's definition of a fast router is one that reports bandwidth above the median of all bandwidth advertisements. A stable router is defined as one that advertises an uptime that is greater than the median uptime of all other routers.

The client will only choose new entry guards when one is unreachable. Currently the default number of entry guards selected is three, and old entry guards that have failed are stored and retried periodically. There is also an option added to use only the entry guards that are hard-coded into the configuration file, but this option is disabled by default. This algorithm was implemented to protect the first hop of a circuit by using a limited pool of nodes.

Non-Entry Router Selection Algorithm. The second algorithm to select non-entry nodes is intended to optimize router selection for bandwidth and uptime, while not always choosing the very best nodes every time. This is meant to ensure that all nodes in the system are used to some extent, but nodes with more bandwidth and higher stability are used most often. Tor has a set of TCP ports that are designated as "long-lived." If the traffic transiting a path uses one of these long-lived ports, Tor will optimize the path for stability by pruning the list of available routers to only those that are marked as stable. This causes Tor's routing algorithm to have a preference towards routers marked as stable nodes. For more details on this part of the algorithm, see the Tor Path Specification [6].

The next part of the algorithm optimizes the path for bandwidth. Briefly, this algorithm works as follows: Let b_i be the bandwidth advertised by the i-th router, and assume that there are N routers. Then the probability that the i-th router is chosen is approximately $b_i \ / \left(\sum_{j=1}^{N} b_j\right)$. We assume that $\sum_{j=1}^{N} b_j > 0$, since a zero value would imply that the system has no available bandwidth. We provide pseudocode for the bandwidth optimization part in Algorithm 1.

The most significant feature of this algorithm is that the more bandwidth a particular router advertises, the greater the probability that the router is chosen. The routing algorithm's tendency to favor stable and high bandwidth nodes is fundamentally important to the implementation of our attack.

Algorithm 1: Non-Entry Router Selection

Input: A list of all known Tor routers, *router_list*

Output: A pseudo-randomly chosen router, weighted toward the routers advertising the highest bandwidth

$B \leftarrow 0$, $T \leftarrow 0$, $C \leftarrow 0$, $i \leftarrow 0$, $router\_bw \leftarrow 0$
$bw\_list \leftarrow \emptyset$
foreach *router* $r \in router\_list$ **do**
  $router\_bw \leftarrow$ get_router_adv_bw(r)
  $B \leftarrow B + router\_bw$
  $bw\_list \leftarrow bw\_list \cup router\_bw$
end
$C \leftarrow$ random_int$(1, B)$
while $T < C$ **do**
  $T \leftarrow T + bw\_list_i$
  $i \leftarrow i + 1$
end
return $router\_list_i$

2.3 Tor's Threat Model

Tor's design document [7] lays out an attack model that includes a non-global attacker that can control or monitor a subset of the network. The attacker can also inject, delay, alter, or drop the traffic along some of the links. This attack model is similar to the models that other low-latency anonymous systems such as Freenet [4], MorphMix [24], and Tarzan [12] are designed to protect against.

As a component of Tor's attack model, the designers acknowledge that an adversary can potentially compromise a portion of the network. To predict the expected percentage of flows compromised by such an adversary, a simplified theoretical analysis of a privacy enhancing system is provided in Tor's design document [7]. This analysis is based on a combinatorial model that assumes nodes are chosen at random from a uniform distribution.

3. COMPROMISING ANONYMITY

We now consider how an adversary might compromise anonymity within the Tor threat model by gaining access to a non-global set of malicious nodes. In our basic attack, we assume that these malicious nodes are fast and stable, as characterized by high bandwidths and high uptimes. While even the basic attack squarely compromises anonymity under Tor's target threat model [7], we also show how to remove these performance restrictions for an even lower-resource attack.

We focus on attacking the anonymity of clients that run in their default configurations; in particular, we assume that clients function only as Tor proxies within the network. We also focus on attacking clients that join the network after the adversary mounts the first phase of our attack (Section 3.1); we shall remove this restriction in Section 5.

3.1 Phase One: Setting Up

To mount our attacks, an adversary must control a subset of $m > 1$ nodes in the pool of active Tor routers. The adversary might obtain such nodes by introducing them directly into the Tor network, or by compromising existing, initially honest nodes. The adversary may coordinate these compromised machines in order to better orchestrate the attack.

The Basic Attack. In our basic attack, the adversary's setup procedure is merely to enroll or compromise a number of high-bandwidth, high-uptime Tor routers. If possible, the adversary should ensure that all of these nodes advertise

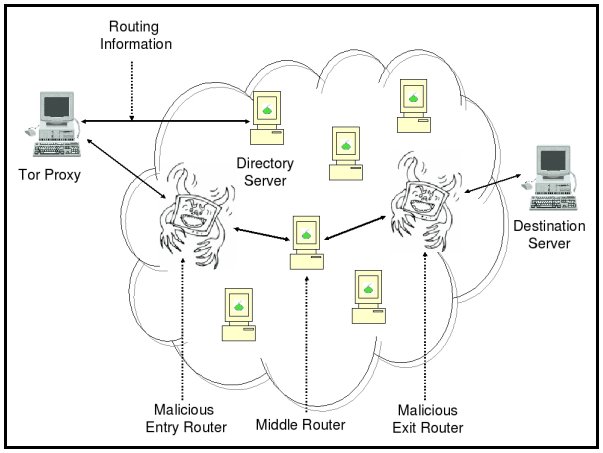

Figure 1: Attack Model: Evil Tor routers are positioned at both the entry and exit positions for a given client's circuit to the requested server through the Tor network.

unrestricted exit policies, meaning that they can forward any type of traffic.

Resource Reduction. We can significantly decrease the resource requirements for malicious nodes, thereby allowing them to be behind low-bandwidth connections, like residential broadband Internet connections. This extension exploits the fact that a malicious node can report incorrect (and large) uptime and bandwidth advertisements to the trusted directory servers [19]. These false advertisements are not verified by the trusted directory servers, nor by other clients who will base their routing decisions on this information, so these false advertisements will remain undetected. Thus, from the perspective of the rest of the network, the adversary's low-resource routers actually appear to have very high bandwidths and uptimes. It is important that the malicious nodes have just enough bandwidth to accept new connections. This is achieved by focusing the nodes' limited resources toward accepting new client connections.

Selective Path Disruption. If malicious nodes do not exist at both the entry and exit positions of a circuit, but at only one position (either entry, middle, or exit), it can cause the circuit to break simply by dropping all traffic along the circuit. This causes the circuit to be rebuilt with a chance that the rebuilding process will create a path configuration in which both the entry and exit nodes are malicious.

What Happens Next. Since one of Tor's goals is to provide a *low-latency* service, when a new client joins the network and initiates a flow, the corresponding Tor proxy attempts to optimize its path by choosing fast and stable Tor routers. By deploying nodes with high bandwidths and high uptimes, or by deploying nodes that give the *impression* of having high bandwidths and high uptimes, the adversary can increase the probability that its nodes are chosen as both entry guards and exit nodes for a new client's circuit. Compromising the entry and exit position of a path is a necessary condition in order for the second phase of our attack (Section 3.2) to successfully correlate traffic.

As a brief aside, on the real Tor network, roughly half of the Tor routers have restricted exit policies that do not allow them to be selected as exit nodes for all flows. This situation further increases the probability that one of the adversary's nodes will be chosen as a flow's exit node.

3.2 Phase Two: Linking Paths

We have shown a method that increases the likelihood of a malicious router existing on a particular proxy's path through Tor. In the improbable case when the full path has been populated with malicious nodes, it is trivial to compromise the anonymity of the path. However, in the more likely case, if only the entry and exit nodes are malicious, we have developed a technique that allows paths to be compromised with a high probability of success (see Figure 1). Our approach here is independent of whether the adversary is implementing the basic or the resource-reduced attack from Section 3.1.

While others have postulated the possibility that an adversary could compromise the anonymity of a Tor route if the adversary controlled both the route's entry and exit nodes [7, 19], to the best of our knowledge, our approach is the first that is capable of doing so *before* the client starts to transmit any payload data. This ability is important, because a resource-starved adversary should desire to minimize the cost of the attack in order to maximize the number of circuits that may be compromised. Furthermore, we experimentally verify the effectiveness of our approach in Section 4.

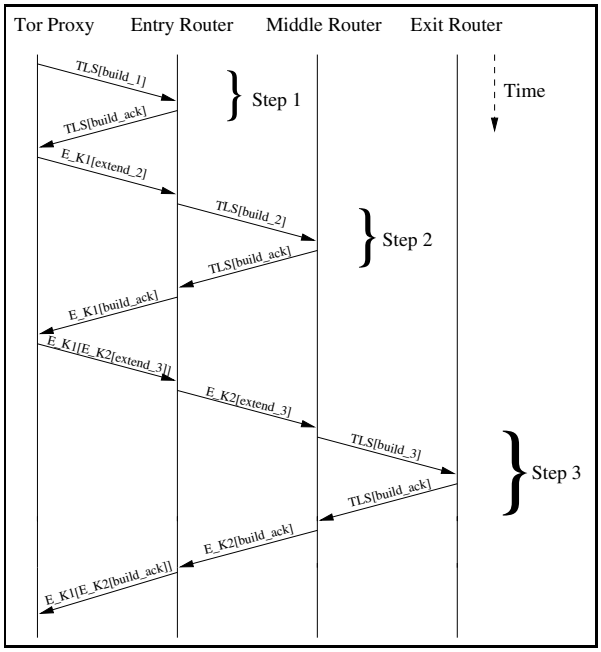

Figure 2: A sequential packet diagram of Tor's circuit building process. In Step 1, the client chooses the first hop along the circuit. Step 2 shows the chosen entry router forwarding the client's request to the chosen middle router. Step 3 shows the entry and middle routers forwarding the final circuit building request message to the desired exit node. Key *K1* is a shared secret key between the client and the entry router. Key *K2* is a shared secret key between the client and the middle router.

Overview. In order for the attack to reveal enough information to correlate client requests to server responses through Tor, each malicious router logs the following infor-

mation for each cell received: (1) its location on the current circuit's path (whether it is an entry, middle, or exit node); (2) local timestamp; (3) previous circuit ID; (4) previous IP address; (5) previous connection's port; (6) next hop's IP address; (7) next hop's port; and (8) next hop's circuit ID. All of this information is easy to retrieve from each malicious Tor router. Once this attack has been carried out, it is possible to determine which paths containing a malicious router at the entry and exit positions correspond to a particular Tor proxy's circuit building requests. With this information, an attacker can associate the sender with the receiver, thus compromising the anonymity of the system. In order to execute this algorithm, the malicious nodes must be coordinated. The simplest approach is to use a centralized authority to which all malicious nodes report their logs. This centralized authority can then execute the circuit-linking algorithm in real-time.

Details. Tor's circuit building algorithm sends a deterministic number of packets in an easily recognizable pattern. Figure 2 shows the steps and the timing associated with a typical execution of the circuit building algorithm. A proxy creates a new circuit through Tor as follows: First, the proxy issues a circuit building request to its chosen entry router and the entry router sends an acknowledgment. Next, the proxy sends another circuit building request to the entry router to *extend* the circuit through a chosen middle router. The middle router acknowledges the new circuit by sending an acknowledgment back to the client via the entry node. Finally, the proxy sends a request to extend the circuit to the chosen exit node, which is forwarded through the entry and middle routers to the chosen exit router. Once the exit router's acknowledgment has been received through the middle and entry nodes, the circuit has been successfully built.

In order to exploit the circuit building algorithm, it is necessary to associate the timing of each step and analyze the patterns in the number and direction of the cells recorded. A packet counting approach as used to locate hidden services [19] would not be sufficient, since not all cells sent from the Tor proxy are fully forwarded through the circuit; thus, the number of cells received at each Tor router along the circuit is different. This pattern is highly distinctive and provides a tight time bound, which we utilize in our circuit linking algorithm.

Our circuit linking algorithm works as follows: The entry node verifies that the circuit request is originating from a Tor proxy, not a router. This is easily determined since there will be no routing advertisements for this node at the trusted directory servers. Next, the algorithm ensures that steps 1, 2, and 3 occur in increasing chronological order. Also, it is necessary to verify that the next hop for an entry node is the same as the previous hop of the exit node. Finally, in step 3, it is verified that the cell headed towards the exit node from the entry node is received before the reply from the exit node. If every step in the algorithm is satisfied, then the circuit has been compromised.

4. EXPERIMENTS

In this section, we describe the experimental process we used to demonstrate and evaluate our resource-reduced attack. By experimentally evaluating our resource-reduced attack, our experimental results also immediately extend to the basic attack scenario in Section 3.

Table 1: Bandwidth Quality Distributions

Tier	Tor Networks		
	Real Tor	40 Node	60 Node
996 KB	38	4	6
621 KB	43	4	6
362 KB	55	6	9
111 KB	140	13	20
29 KB	123	11	16
20 KB	21	2	3
Total	103.9 MB	10.4 MB	15.7 MB

4.1 Experimental Setup

In order to evaluate this attack in a realistic environment, we set up an isolated Tor deployment on the PlanetLab overlay testbed [22]. We were advised not to validate our attack on the real Tor network because of its potentially destructive effect [5]; however, we did verify that our technique for publishing false router advertisements did, in fact, propagate for a single test router on the real Tor deployment.

To ensure that the experimental Tor networks are as realistic as possible, we surveyed the real Tor network in August 2006 to determine the router bandwidth distribution. This data is given in Table 1. According to the real trusted Tor directory servers, there are roughly 420 Tor routers in the wild that forward at least 5 KB per second. However, due to limitations on the number of PlanetLab nodes that were available over the course of the experiments, we created smaller Tor networks according to our analysis of the router quality distribution in the real deployment. We created two isolated Tor networks on PlanetLab, consisting of 40 and 60 nodes, each running exactly one router per node. Each experimental deployment has precisely three directory servers, which are also nodes from PlanetLab.

When choosing nodes from PlanetLab for the experimental deployments, each node was evaluated using `iperf`, a common bandwidth measurement tool [15], to ensure that it had sufficient bandwidth resources to sustain traffic at its assigned bandwidth class for the course of each experiment. Also, as is consistent with the real Tor network, we chose PlanetLab nodes that are geographically distributed throughout the world.

All Tor routers (both malicious and benign) advertise the same, unrestricted exit policy. The exit policies of routers in the real Tor network are difficult to accurately model due to the reduced size of our network. The global use of unrestricted exit policies in our experimental Tor testbed actually decreases the potential effectiveness of our attack. With the potential for more restrictive exit policies in a real Tor network, we expect the attack's performance to improve since the malicious routers would have a higher probability of compromising the exit positions.

To demonstrate the effect of the attack, we introduced a small number of malicious Tor routers into each private Tor network. In the 40 node network, experiments were conducted by adding two (2/42) and four (4/44) malicious nodes. In the 60 node network, three (3/63) and six (6/66) malicious nodes are added. The fraction of each network's bandwidth that is malicious is given in Section 4.3. All experiments were conducted in October 2006 with `Tor` version 0.1.1.23.

The experiments were conducted as follows: The three trusted directory servers and each benign Tor router in the

network are started first, then the client pool begins generating traffic through the network. The network is given two hours for routing to converge to a stable state,[1] at which point the clients are promptly stopped and all previous routing information is purged so that the clients behave exactly like new Tor proxies joining the network. The malicious nodes are then added to the network and the clients once again generate traffic for precisely two hours. This procedure is repeated for the 2/42, 4/44, 3/63, and 6/66 experiments. The results of these experiments are given in Section 4.4.

4.2 Traffic Generation

To make the experimental Tor deployments realistic, it is necessary to generate traffic. Unfortunately, there is not much data available on the nature of Tor traffic and exact numbers of clients on the real Tor network. Therefore, we adopt the same traffic-generation strategy as Murdoch [16]. To generate a sufficient amount of traffic, we used six dual Xeon class machines running GNU/Linux with 2GB of RAM on a 10 Gbit/s link running a total of 60 clients in the 40 node network, and a total of 90 clients in the 60 node Tor deployment. These clients made requests for various web pages and files of relatively small size (less than 10 MB) using the HTTP protocol. The interface between the HTTP client and the Tor proxy is made possible by the `tsocks` transparent SOCKS proxy library [27]. The clients also sleep for a random period of time between 0 and 60 seconds and restart (retaining all of their cached state including routing information and entry guards) after completing a random number of web requests so that they do not flood the network.

4.3 Malicious Node Configuration

To maximize the amount of the anonymized traffic that an attacker can correlate, each malicious router advertises a read and write bandwidth capability of 1.5 MB/s and a high uptime. Furthermore, each malicious node is rate limited to a mere 20 KB/s for both data and controls packets to make the node a low-resource attacker. In terms of the total network bandwidth in each deployment, the addition of malicious nodes contribute a negligible amount of additional bandwidth. In the 2/42 and 3/63 experiments, the malicious nodes comprise 0.38% of each network's bandwidth while actually advertising approximately 22% of the total bandwidth. In the 4/44 and 6/66 experiments, malicious nodes make up 0.76% of the bandwidth while advertising about 36% of the total network's bandwidth.

In addition, each malicious node logs the necessary information for the path linking algorithm, as described in Section 3.2. The malicious routers' behavior is aimed at maximizing the probability that it will be included on a circuit.

4.4 Experimental Results

In this section, we present the results of the experiments on our isolated Tor deployments. To demonstrate the ability of the attack to successfully link paths through the Tor network, we measured the percentage of the Tor circuits that our path linking algorithm (Section 3.2) can correctly correlate.

[1]Our attempts to start the network with both honest and malicious nodes at once failed, due to the inability of the honest nodes to integrate into the hostile network. The two hour time period allowed the honest nodes time to fully integrate into the routing infrastructure before adding the malicious nodes.

Table 2: The raw number of compromised circuits

	Number of Circuits	
	Compromised	Total
2/42	425	4,774
4/44	3,422	10,199
3/63	535	4,839
6/66	6,291	13,568

Using the data logged by malicious routers, our path linking algorithm was able to link a relatively high percentage of paths through Tor to the initiating client. In the 40 Tor router deployment, we conducted experiments by adding two (2/42) and four (4/44) malicious nodes. The malicious routers composed roughly 4.8% and 9.1% of each network, or 0.38% and 0.76% of each network's bandwidth. In the 2/42 experiment, the malicious nodes were able to compromise approximately 9% of the 4,774 paths established through the network. We then performed the 4/44 experiment, and were able to correlate approximately 34% of the 10,199 paths through the network. Thus, the attack is able to compromise the anonymity of over one-third of the circuit-building requests transported through the experimental network.

These experiments are repeated for a network of 60 Tor routers by adding three (3/63) and six (6/66) malicious nodes. The malicious routers composed about 4.8% and 9.1% of each network, or 0.38% and 0.76% of each network's bandwidth. With only three (3/63) malicious routers, the attack compromises about 11% of the 4,839 paths and in an experiment with six (6/66) malicious Tor routers, the attack compromised over 46% of the 13,568 paths. The results as percentages of compromised paths are given in Tables 3 and 4. The raw number of compromised circuits in each experiment is given in Table 2.

In addition to the correctly correlated paths, there were only 12 incorrectly correlated paths over all the experiments (one false positive in the 3/63 experiment, three in the 4/44 experiment, and eight in the 6/66 experiments). The negligible number of false positives shows that our path linking algorithm is highly accurate; however, the low false-positive rate may also be a result of the relatively light and uniform traffic load that was generated.

Table 3: The number of predicted and actual circuits compromised in the 40 node PlanetLab network.

	Experiments	
	2/42	4/44
Random Selection	0.12%	0.63%
Experimental	8.90%	33.55%
Improvement	7,565%	5,190%

In Tables 3 and 4, the experimental results are compared to an analytical expectation of the percentage of paths that can be compromised by controlling the entry and exit nodes if routers are selected uniformly at random. The analytical expectation is based on a combinatorial model originally defined in Tor's design document [7] as $(\frac{m}{N})^2$, where $m > 1$ is the number of malicious nodes and N is the network size (at its inception, Tor did not provide load-balancing; routers were selected uniformly at random). This analytical model does not take into account the fact that a Tor router may

Table 4: The number of predicted and actual circuits compromised in the 60 node PlanetLab network.

	Experiments	
	3/63	6/66
Random Selection	0.15%	0.70%
Experimental	11.06%	46.36%
Improvement	7,097%	6,530%

be used only once per circuit. Thus, a more precise expectation can be described by $(\frac{m}{N})(\frac{m-1}{N-1})$, $m > 1$. The predicted fraction of compromised circuits if routers were chosen at random is given in Tables 3 and 4.

The distinction between the uniform selection expectations and the experimental results is clear; the experiments demonstrate that Tor's addition of load balancing has caused the number of circuits compromised to increase by 52 and 76 times over uniformly random router selection.

4.5 Attack Discussion

It is worth asking why the earlier analytical model based upon uniform router selection that predicted a strong resistance to this type of attack does not match our experimental results. Besides enabling malicious nodes to lie about their resources, the primary issue is that the analytical model assumes resource homogeneity across the set of Tor nodes, when in fact the real Tor network has a heterogeneous resource distribution (see Table 1). As a result, routers are not chosen with an equal probability; those with higher bandwidth claims are chosen more frequently. Also, our attack used selective path disruption to cause circuits to fail, which is not considered in the analytical model.

Furthermore, since routers have finite resources, they must reject new connections once their resources are consumed. To make matters worse, most of the available bandwidth from low-resource nodes may be exhausted by protocol control overhead, which decreases the effective size of the network while increasing the probability of a malicious node being selected. This means that if the Tor network becomes heavily congested, it would magnify the effectiveness of our attack.

Given the fraction of bandwidth necessary to compromise a significant number of circuits in the experimental networks, it is possible to extrapolate an estimate of the attack's performance in larger Tor networks. Attackers contributing less than 1% (between 0.38% and 0.78%) of the network's aggregate bandwidth were able to compromise up to 46% of the circuit-building requests for new Tor proxies. Now, assume a Tor network similar in bandwidth distribution to the real Tor deployment in August 2006 (see Table 1) with 104 MB of total bandwidth. If an adversary could contribute an upper bound of an additional 1% of bandwidth, only 1.04 MB/s, then one could expect the attack to compromise a similar fraction of circuit requests as in our experiments.

However, this analysis remains an open problem, due to the variable router quality, the increasing size of the real Tor network, the correctness of our traffic generation model, and the Tor Project's request [5] that we not experiment with our attacks on the real Tor network.

5. ATTACK EXTENSIONS

Having presented the basic ideas behind our attacks, we consider further attack variants and improvements, such as attacking existing Tor clients instead of only new Tor clients, router advertisement flooding, and watermarking attacks.

Compromising Existing Clients. Clients that exist within the network before the malicious nodes join will have already chosen a set of entry guard nodes. We present two methods to compromise the anonymity of existing clients. First, if an attacker can observe the client (e.g., by sniffing the client's 802.11 wireless link), he/she can easily deduce the entry guards used by a particular client. The adversary can then make those existing entry guards unreachable or perform a denial-of-service (DoS) attack on these entry guards, making these nodes unusable. This forces the client to select a new list of entry guards, potentially selecting malicious Tor routers. Another method to attack clients that have a preexisting list of entry guard nodes would be to DoS a few key stable nodes that serve as entry guards for a large number of clients. This would cause existing clients to replace unusable entry guards with at least one new and potentially malicious entry guard node.

Improving Performance Under the Resource Reduced Attack. One concern with the resource-reduced attack that we describe in Section 3 is that, by itself, the attack can seriously degrade the performance of new Tor clients. The degradation in performance could then call attention to the malicious Tor nodes. Naturally, the basic attack in Section 3 would be completely indistinguishable from a performance perspective since the basic adversary does not lie about its resources.

The first question to ask is whether poor performance under an adversarial situation is a sufficient protection mechanism. We believe that the answer to this question is "no" — it is a poor design choice for users of a system to have to detect an attack based on poor performance. A better approach is to have an automated mechanism in place to detect and prevent our low-resource attack. Furthermore, a resource-reduced adversary could still learn a significant amount of private information about Tor clients between the time when the adversary initiates the attack and time when the attack is discovered.

The second, more technical question is to ask what a resource-reduced adversary might do to improve the perceived performance of Tor clients. One possible improvement arises when the attacker wishes to target a particular client. In such a situation, the adversary could overtly deny service to anyone but the target client. Specifically, an adversary's Tor nodes could deny (or passively ignore) all circuit-initiation requests except for those requests that the target client initiates. This behavior would cause the non-target clients to simply exclude the adversary's nodes from their lists of preferred entry guards, and would also prevent non-target clients from constructing circuits with the adversary's nodes as the middle or exit routers. Since circuit-formation failures are common in Tor [20], we suspect that this attack would largely go unnoticed.

Displacing Honest Entry Guards. Recall that Tor uses special entry guard nodes to protect the entry of a circuit from selective disruption. In order to be marked by the directory servers as a possible entry guard, a Tor router must advertise an uptime and bandwidth greater than the median advertisements (in `Tor` version 0.1.1.23). Another attack,

which is a variant of the Sybil attack [10], can be conducted by flooding the network with enough malicious routers advertising high uptime and bandwidth. On our isolated Tor network, we successfully registered 20 routers all on a single IP address and different TCP port numbers.[2] Flooding the network with false router advertisements allows a non-global adversary to effectively have a "global" impact on Tor's routing structure. Namely, this attack increases the median threshold for choosing entry guards, thereby, preventing benign nodes from being marked as potential entry guards. This attack could help guarantee that only malicious nodes can be entry guards.

Compromising Only the Entry Node. As another extension to our attack, suppose that an adversary is less interested in breaking anonymity in general, but is instead particularly interested in correlating Tor client requests to a specific target website (such as a website containing controlled or controversial content). Suppose further that the adversary has the ability to monitor the target website's network connection; here the adversary might have established the target website to lure potential clients, or might have obtained legal permission to monitor this link. Under this scenario, an adversary only needs to compromise an entry node in order to correlate client requests to this target website. The critical idea is for the entry router to watermark a client's packets using a time-based watermarking technique, such as the technique used in Wang, *et al.* [28] or other variants. The adversary's malicious entry routers could embed a unique watermark for each client-middle router pair. A potential complication might arise, however, if the client is using Tor to conceal simultaneous connections to multiple websites, and if the circuits for two of those connections have the same middle router.

6. PROPOSED DEFENSES

Non-global, but high-resource (uptime, bandwidth), adversaries seem to pose a fundamental security challenge to any high-performance, multi-hop privacy enhancing system that attempts to efficiently balance its traffic load, and we welcome future work directed toward addressing this challenge. We consider, however, methods for detecting non-global *low-resource* adversaries.

In order to mitigate the negative effects of false routing information in the network, it is necessary to devise a methodology for verifying a router's uptime and bandwidth claims. Here, we provide a brief overview of some potential solutions and alternative routing schemes.

6.1 Resource Verification

Verifying Uptime. A server's uptime could be checked by periodically sending a small heartbeat message from a directory server. The additional load on the directory server would be minimal and it could effectively keep track of how long each server has been available.

Centralized Bandwidth Verification. Since Tor relies upon a centralized routing infrastructure, it is intuitive to suggest that the trusted centralized directory servers, in addition to providing routing advertisements on behalf of Tor routers, also periodically verify the incoming bandwidth advertisements that are received from Tor routers. The directory server could measure a router's bandwidth before publishing the routing advertisement and correct the value if it found that the router does not have its claimed bandwidth. The difficulty with this approach is that it cannot detect selectively malicious nodes. Therefore, it is necessary for the bandwidth verification mechanism to continuously monitor each node's bandwidth. Due to the significant traffic load that would be placed upon the few centralized directory servers, this centralized bandwidth auditing approach would create a significant performance bottleneck.

Distributed Bandwidth Verification. In order to detect false bandwidth advertisements, it may be tempting to augment the routing protocol to allow Tor routers to proactively monitor each other. Anonymous auditing [25], where nodes anonymously attempt to verify other nodes' connectivity in order to detect collusion, has been proposed as a defense against routing attacks in structured overlays. A similar technique could be designed to identify false resource advertisements. However, this technique is also insufficient at detecting selectively malicious nodes. In addition, this approach introduces additional load in the network and could result in significant performance degradation.

Distributed Reputation System. Reputation systems have been proposed for anonymous systems within the context of reliable MIX cascades [9]. One could envision a reputation system similar to TorFlow [21], that actively verifies the directory server reported bandwidth claims for each Tor router and dynamically updates each bandwidth claim. Selectively malicious nodes are still difficult to detect with such a reputation system.

6.2 Mitigating Sybil Attacks

In order for any reputation system to be effective, it is necessary to address the Sybil attack [10]. Recall that the directory servers place no constraints upon the number of Tor routers that may exist at a single IP address (as of `Tor` version 0.1.1.23). This can be exploited to effectively replace all entry guards with malicious nodes (see Section 5). To help mitigate this kind of attack, the directory servers should limit the number of routers introduced at any single IP address. Furthermore, enforcing location diversity increases the resources required to perform this attack [11]. Following the initial disclosure our results [2], Tor adopted countermeasures that (1) allow only three Tor routers to be hosted at any single IP address and (2) dictate that circuits may not include more than one router from a particular class B address space [1].

6.3 Alternative Routing Strategies

Since Tor's desire to efficiently balance its traffic over the available resources in the network has left it vulnerable to traffic correlation attacks, it is prudent to consider alternate routing strategies that may provide adequate load balancing while preserving the network's anonymity.

Proximity Awareness. Secure routing based on *proximity awareness* has been proposed in peer-to-peer networks [3]. In such a routing strategy, the next hop is computed by minimizing a distance metric, such as round trip time (RTT). Proximity-based routing may over-optimize and cause circuits to be built deterministically. Also since paths are multi-hop and source routed, the client would need distance

[2]The trusted directory servers currently (as of `Tor` version 0.1.1.23) have no limits as to the number of routers that can be hosted on a single IP address. In theory, an attacker can register up to $2^{16} - 1$ Tor routers to the same IP address.

metrics for the first hop to the second hop and the second hop to the third hop. In addition, these metrics must be verified. Finally, proximity routing seems to be incompatible with enforcing location diversity.

Loose Routing. Loose routing in anonymous systems has been proposed in Crowds [23], where path lengths are non-deterministic since each hop chooses to forward to another intermediate hop probabilistically. This strategy places a great amount of trust on the entry nodes. A malicious entry node could simply route all traffic immediately to the exit server.

Local Reputation-based Routing. Another scheme could be to initially choose paths with a uniform probability and over time, maintain local reputation information for all nodes used in a path. At the start, the performance would be expectedly poor, but over time, as clients begin to choose high quality Tor routers, they can begin to optimize for performance. This approach is not vulnerable to false advertisements.

We plan to explore these alternate routing strategies as future work, in particular, focusing on an analysis of the anonymity that each respective routing technique provides.

7. CONCLUSION

We present a low-resource end-to-end traffic analysis attack against Tor that can compromise anonymity before any payload data is sent. The attack stems from Tor's tendency to favor routers that claim to be high-resource with high-uptime in its routing process in an attempt to optimally balance the traffic load. However, since there is no mechanism to verify resource claims, we experimentally show that it is possible for even a low-resource adversary to compromise an unfairly large fraction of the circuit-building requests through the network.

In addition, we illustrate the feasibility of displacing all entry guard nodes with malicious nodes, thereby having a global effect upon the Tor's routing mechanism. Attack extensions are presented that further reduce the cost of launching this atack.

To mitigate the low-resource variety of these attacks, we propose solutions aimed at verifying all bandwidth and uptime claims. However, these attacks highlight the inherent challenge in designing an anonymity-preserving reputation system that is robust to a selectively malicious adversary.

Since this study shows how anonymity and efficient resource (bandwidth) use appear to be in opposition, our hope is that these attacks motivate further research in the area of designing and implementing optimal routing algorithms in anonymous overlay networks that deliver a high level of performance without compromising the security of any aspect of the system.

Acknowledgements. We thank Nikita Borisov, Roger Dingledine, Paul Syverson, and the anonymous reviewers whose insightful feedback greatly improved the quality of this paper. This research was partially funded by NSF Award 0430593, "ITR: Privacy and Surveillance In Wireless Networks" and NSF CRI award 0454404 "CRI: Wireless Internet Building Blocks for Research, Policy, and Education." T. Kohno was supported in part by NSF Grant CNS-0627157 and gifts from Cisco and Intel.

8. REFERENCES

[1] Bauer, K., and McCoy, D. Tor specification proposal 109: No more than one server per ip address. `http://tor.eff.org/svn/trunk/doc/spec/proposals/109-no-sharing-ips.txt`, March 2007.

[2] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. Low-resource routing attacks against anonymous systems. Computing Science Technical Report CU-CS-1025-07, University of Colorado, Feb. 2007.

[3] Castro, M., Druschel, P., Ganesh, A., Rowstron, A., and Wallach, D. S. Secure routing for structured peer-to-peer overlay networks. In *OSDI 2002*.

[4] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability* (2000).

[5] Dingledine, R. Personal communication., October 2006.

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

[7] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *13th USENIX Security Symposium* (2004).

[8] Dingledine, R., Mathewson, N., and Syverson, P. Challenges in deploying low-latency anonymity. NRL CHACS Report 5540-625, 2005.

[9] Dingledine, R., and Syverson, P. Reliable MIX Cascade Networks through Reputation. In *Proceedings of Financial Cryptography (FC 2002)*.

[10] Douceur, J. The Sybil Attack. In *Proceedings of International Peer To Peer Systems Workshop (IPTPS 2002)*.

[11] Feamster, N., and Dingledine, R. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)* (2004).

[12] Freedman, M. J., and Morris, R. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)* (Washington, DC, November 2002).

[13] Goldberg, I. On the security of the tor authentication protocol. In *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)* (Cambridge, UK, June 2006), Springer.

[14] Goldschlag, D. M., Reed, M. G., and Syverson, P. F. Hiding Routing Information. In *Proceedings of Information Hiding: First International Workshop* (May 1996), Springer-Verlag, LNCS 1174, pp. 137–150.

[15] Iperf - The TCP/UDP Bandwidth Measurement Tool. `http://dast.nlanr.net/Projects/Iperf`.

[16] Murdoch, S. J. Hot or not: Revealing hidden services by their clock skew. In *13th ACM Conference on Computer and Communications Security (CCS 2006)* (Alexandria, VA, November 2006).

[17] Murdoch, S. J., and Danezis, G. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy* (May 2005), IEEE CS.

[18] Murdoch, S. J., and Zieliński, P. Sampled traffic

analysis by internet-exchange-level adversaries. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2007)* (June 2007).

[19] Øverlier, L., and Syverson, P. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy* (May 2006), IEEE CS.

[20] Perry, M. Securing the tor network. Defcon 2007. `http://fscked.org/transient/SecuringTheTorNetwork.pdf`.

[21] Perry, M. Torflow. `http://tor.eff.org/svn/torflow/README`.

[22] Peterson, L., Muir, S., Roscoe, T., and Klingaman, A. PlanetLab Architecture: An Overview. Tech. Rep. PDN–06–031, PlanetLab Consortium, May 2006.

[23] Reiter, M., and Rubin, A. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security 1*, 1 (June 1998).

[24] Rennhard, M., and Plattner, B. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)* (Washington, DC, USA, November 2002).

[25] Singh, A., Druschel, P., and Wallach, D. S. Eclipse attacks on overlay networks: Threats and defenses. In *IEEE INFOCOM* (2006).

[26] Syverson, P., Tsudik, G., Reed, M., and Landwehr, C. Towards an Analysis of Onion Routing Security. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (July 2000), H. Federrath, Ed., Springer-Verlag, LNCS 2009, pp. 96–114.

[27] Transparent SOCKS Proxying Library. `http://tsocks.sourceforge.net`.

[28] Wang, X., Chen, S., and Jajodia, S. Tracking anonymous peer-to-peer voip calls on the internet. In *Proceedings of the ACM Conference on Computer and Communications Security* (November 2005), pp. 81–91.