Low-Resource Routing Attacks Against Anonymous Systems

Kevin Bauer[1] Damon McCoy[1] Dirk Grunwald[1] Tadayoshi Kohno[2] Douglas Sicker[1]

[1]University of Colorado, Boulder, CO, USA [2]University of Washington, Seattle, WA, USA

University of Colorado at Boulder
Technical Report CU-CS-1025-07

Abstract

Overlay mix-networks are widely used to provide low-latency anonymous communication services. It is generally accepted that, if an adversary can compromise the endpoints of a path through an anonymous mix-network, then it is possible to ascertain the identities of a requesting client and the responding server. However, theoretical analyses of anonymous mix-networks show that the likelihood of such an end-to-end attack becomes negligible as the network size increases. We show that if the mix-network attempts to optimize performance by utilizing a preferential routing scheme, then the system is highly vulnerable to attacks from non-global adversaries with only a few malicious servers.

We extend this attack by exploring methods for low-resource nodes to be perceived as high-resource nodes by reporting false resource claims to centralized routing authorities. To evaluate this attack on a mature and representative system, we deployed an isolated Tor network on the PlanetLab testbed. We introduced low-resource malicious nodes that falsely gave the illusion of high-performance nodes, which allowed them to be included on a disproportionately high number of paths. Our results show that our malicious low-resource nodes are highly effective at compromising the end-to-end anonymity of the system. We present several extensions to this general attack that further improve the performance and minimize the resources required. In order to mitigate low-resource nodes from exploiting preferential routing, we present several methods to verify resource claims, including a distributed reputation system. Our attacks suggest what seems be a fundamental problem in multi-hop systems that attempt to simultaneously provide anonymity and high-performance.

1 Introduction

Overlay mix-networks are widely used to provide low-latency anonymous communication services. It is generally accepted that it is impossible for any practical privacy enhancing system to provide perfect anonymity. Therefore, the designers of such systems must consider restricted threat models. In particular, many systems acknowledge that, if the endpoints of a path through the mix-network are compromised, it is possible to perform a traffic analysis attack. However, theoretical analyses of anonymous mix-networks show that the likelihood of successfully launching such a traffic analysis attack becomes negligible as the network size increases [9, 24, 29].

In the case of *Tor* [9], one of the most popular privacy enhancing systems, the explicitly stated goal is that the system should provide anonymity against *non-global adversaries*. Although, in theory, such a system is resistant to end-to-end traffic analysis attacks, our results show quite the contrary. Since Tor attempts to provide optimal performance for interactive applications, it employs a preferential routing mechanism. We show that even if an adversary can only control a *few* malicious nodes — 3 or 6 in our experiments with a PlanetLab network of 60 honest servers — the adversary can still compromise the anonymity of a significant fraction of the connections from new clients. Moreover, since our attacks exploit Tor's preferential routing algorithms, which are critical to ensuring it's high-performance, our results have broad implications to all high-performance, low-latency anonymity systems.

Research Themes. Our attacks enable both positive and negative adversarial actions, depending on who one considers to be the adversary. For example, law enforcement officers might use our techniques to cheaply and realistically track online child predators, and the RIAA and other organizations might use our techniques to link web or torrent requests to their corresponding requesters. Rather than focus on particular applications, however, we focus our research on the following two basic 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.

Attack Overview. All of our attacks extend the following very simple insight: Since Tor's routing mechanism prefers high-bandwidth, high-uptime servers for certain portions of a flow's route, an adversary can bias the route selection algorithm toward malicious nodes with high bandwidths and high uptimes. The adversary only needs a few such malicious nodes in order to successfully compromise the anonymity of a significant number of clients. In this basic attack, an adversary could deploy a few high-bandwidth, high-uptime servers, and with high probability, compromise two of the critical Tor servers on a Tor route for new Tor clients; technically, the compromised nodes are the *entry* and *exit* nodes.

Compromising Anonymity by Linking Paths. We present a new end-to-end method for associating a client's request to its corresponding destination server. After the attacker has compromised the entry and exit nodes, they can use our *path linking algorithm* to bind together both parties in the flow. Once the identities of the sender and receiver are discovered, the system's anonymity has been fully compromised.

This new flow linking algorithm is the first such technique that that can link flows *before* any payload data is transmitted. An advantage to linking a flow before any payload data is sent is that if the adversary is unable to link the path, but does control at least one router along the path, than the adversary can at least terminate the path before it is fully constructed. Doing so would force the client to initiate a new path construction request, thereby giving the adversary another opportunity to link the client with the target server.

Reducing Per-Node Resource Requirements. While we consider the basic attack to be quite serious, we can do significantly better. Namely, while it only requires the adversary to deploy a few malicious Tor nodes, the basic attack still mandates that those nodes have high bandwidth connections and high uptimes. As our next contribution, we show that even adversaries with sparse resources — such as adversaries with a few nodes behind residential cable modems — can compromise the anonymity of many paths. Our extension to low-resource adversaries exploits the fact that a node can lie about it's resources, since Tor's routing infrastructure does *not* verify a server's resource claims.

To sample our results, in one of our experiments, we constructed an isolated Tor network on the PlanetLab testbed consisting of 60 honest nodes and 6 malicious nodes falsely claiming to have high bandwidths and high uptimes. In this configuration, we introduced 90 new clients that issued approximately 13,500 HTTP requests over the course of two hours. By applying our path linking algorithm and resource reduction techniques, the adversary could compromise over 46% of the paths in the network. This is in stark contrast to the 0.70% of paths predicted by the previous analytical model [9].

Context. It is worth asking why the earlier theoretical model [9] predicting strong resistance to this type of attack does not match our experimental results. Besides allowing malicious nodes to lie about their resources, the main issue is that the theoretical model assumes a homogeneous set of Tor nodes, when in fact a real Tor network will be vastly heterogeneous. While the Tor developers seem to have realized that the previous analytical model does not reflect the full complexities of a real deployment [8], we are the first to experimentally analyze and push the limits of the practical implications of Tor's heterogeneous architecture on it's anonymity. In addition, it is important to note that, since Tor uses a centralized method to maintain and distribute the full list of routers, it is not vulnerable to many of the routing attacks that are possible in decentralized overlay systems. These include, for example, the Eclipse attack [28], attacks on distributed hash tables (DHTs) [1], and passive node profiling attacks [4].

Additional Improvements and Other Attacks. 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. Additionally, we consider methods to improve the effectiveness of our attack, such as a variant of the Sybil attack [10].

While one might envision additional attacks against Tor, such as enrolling each member of a large botnet as a Tor server, or enrolling a large number of virtual machines as Tor servers, we view such attacks as orthogonal to our research since our fundamental goal is to reduce the resource requirements on the adversary, not to suppose additional resources.

Towards Prevention. 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 prevent Sybil attacks.

Broader Implications. Our results suggest what might be a fundamental problem with any multi-hop system that tries to simultaneously provide anonymity and high-performance. Indeed, our results suggest that one must be extremely careful when optimizing a system for high performance since, if an adversary can place high-performance servers in the network, or give the illusion of high-performance servers, then a solution optimized for high-performance could allow the adversary to compromise the anonymity of many users.

Outline. The remainder of this paper is organized as follows: In Section 2 we present a brief overview of the Tor system, an in depth analysis of Tor's router selection algorithms, a description of Tor's envisioned attack model, and an overview of the common anonymity metrics that are used throughout the remainder of the paper. In Section 3, we introduce our attack and describe our attack model. Section 4 provides a description of the experiments that are used to validate our attack and we discuss the experimental results. In Section 5, we present possible extensions to our attack. Section 6 offers a detailed analysis of our proposed defenses, and in Section 7, we compare our attack with previous attacks against Tor. Section 8 concludes with a discussion of the implications of our results on the design and implementation of privacy enhancing systems.

2 Background

In order to present our attack methodology, we provide a brief overview of the Tor system, an in depth analysis of Tor's router selection algorithms, a description of Tor's envisioned attack model, and an overview of the common anonymity metrics that are used throughout the remainder of the paper.

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 high-throughput and low-latency mix network [2].

In the Tor architecture, there are several fundamental concepts which are defined as follows (see Figure 1): An *onion router* (OR) is the server component of the network that is responsible for forwarding traffic within the core of the mix network. An *onion proxy* (OP) is the client part of the network that injects the user's traffic into the network of onion routers; for our purposes, one can view the onion proxy as a service that runs on the user's computer. A *circuit* is a path of three onion routers (by default) through the Tor network from the onion proxy to the desired destination server. The first onion router on the circuit is re-

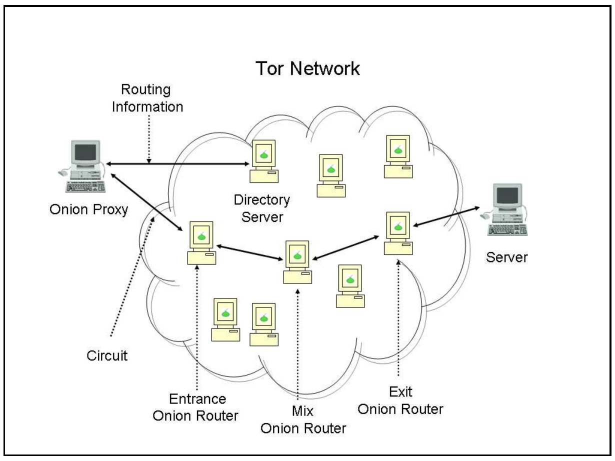

Figure 1: **Tor Overview**

ferred to as the *entrance* router, the second router is called a *mix* router, and the final hop is the *exit* router. Onion proxies choose stable and high bandwidth onion routers to be *entry guards*, which are used as an entrance router. We use the terms *entrance guard* and *entrance 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 Tor network is called a *cell*, which is a fixed-size 512 byte packet, that is padded if necessary.

Tor's low-latency objective requires that the network design make some trade-offs between performance and anonymity. In particular, Tor does not explicitly delay or re-order cells. A separate set of buffers are created to store cells received from each circuit. Cells are forwarded using a round-robin queuing model to give a fair amount of bandwidth to each circuit and to minimize latency.

At the core of Tor is a circuit switched network. The circuits are carefully built to ensure that only the originator of the circuit knows the complete path through which the cell is routed. Tor also uses Transport-Layer Security (TLS) [6] and other standard cryptographic methods to prevent an eavesdropper from linking incoming cells to outgoing cells. 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 its final destination. This process is known as *onion routing* [14]. It is important to note that only the exit node can observe the unencrypted contents of the cell. Since our attack does not focus on the cryptographic security of any of these protocols, we will not go into any more detail. 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 [9].

Tor can operate as both an onion proxy, which builds circuits to forward a local user's traffic through the network and also as an onion 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 an onion proxy (client), handling only the local user's traffic.

2.2 Tor's Router Selection Algorithm in Detail

There are currently two parts to the algorithm that Tor uses to select which onion routers to include in a circuit. The first part is used to select the entrance 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.

Entrance Router Selection Algorithm. The default algorithm used to select entrance routers was modified in May 2006 with the release of Tor version 0.1.1.20. Entry guards were introduced to protect the beginning of circuits [22]. The entry guard selection algorithm works by automatically selecting a set of onion 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 hardcoded 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 what are believed to be more reliable and trustworthy nodes.

Non-Entrance Router Selection Algorithm. The second algorithm to select non-entrance nodes is intended to optimize onion 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 onion routers marked as stable nodes. For more details on this part of the algorithm, see the Tor Path Specification [8].

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 $\sum_{j=1}^{N} b_j$ is positive, since a zero value would imply that the system has no available bandwidth. We provide pseudocode for the bandwidth optimization part of the algorithm in Appendix A. 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.

2.3 Tor's Threat Model

Tor's design document [9] 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 attack model that other low-latency anonymous systems such as Freenet [3], MorphMix [25], 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. This analysis is based on a combinatorial model that assumes nodes are chosen at random from a uniform distribution. The designers make the assumption that, due to the low-latency nature of Tor, it is possible to correlate a traffic flow with only the *entrance* and *exit* nodes compromised (we agree in this regard, and provide a new method for doing so in Section 3). According to the designers, the probability of choosing a compromised entrance node is $\frac{m}{N}$ and the probability of choosing a compromised exit node is the same, thus, the combinatorial model is expressed as $(\frac{m}{N})^2$, where $m > 1$ is the number of malicious nodes and N is the network size [9].

2.4 Common Anonymity Metrics

Using *entropy* as a metric to measure the amount of anonymity provided by a system was proposed in Diaz, *et al.* [5], and Serjantov and Danezis [26]. From Shannon's information theory [27], the notion of entropy is defined as follows:

$$H(N) = -\sum_{x_i \in N} p(x_i) \log_2 (p(x_i));$$

where $p(x_i)$ is the probability of the i-th node being included on a path and N is the set of all routers in the Tor network. In Zhuang, *et al.* [33], a metric is given to measure the *unlinkability* of a system based on the entropy calculation. The ideal entropy of a system is calculated by $I(N) = \log_2 (|N|)$. The unlinkability calculation normalizes the raw entropy of a system by dividing by the ideal entropy, $\frac{H(N)}{I(N)}$. This metric can show how well a system does at making it appear equally likely that any user in the system could have performed an action. Even with this metric, anonymity is still a fuzzy concept that is hard to fully measure, since a high entropy value in a system is a necessary, but not sufficient, condition for strong anonymity. A low entropy value in a system indicates that it does not provide strong anonymity. However, when measuring the anonymity of a system, one must also take into account if the system is vulnerable to common attacks (or previously undocumented attacks), that degrade the anonymity of the system.

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 [9], 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 onion 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 onion 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 our 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 servers. If possible, the adversary should ensure that all of these nodes advertise unrestricted exit policies, meaning that they can forward any type of traffic.

Resource Reduction. We can significantly decrease the resource requirements for the 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. 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 servers actually appear to have very high bandwidths and uptimes.

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 onion proxy attempts to optimize its path by choosing fast and stable onion 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 entrance 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 onion 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 entrance and exit nodes are malicious, we have developed a technique that allows paths to be compromised with a high probability of success (see Figure 2). 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 [9], to the best of our knowledge, ours is the first

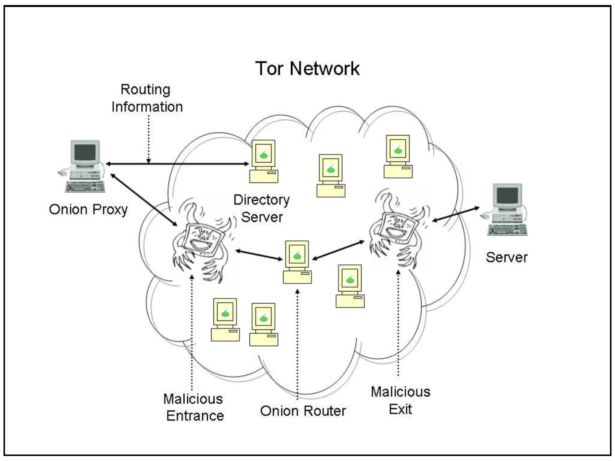

Figure 2: **Attack Model**: Evil onion routers are positioned at both the entrance and exit positions for a given client's path to the requested server through the Tor network.

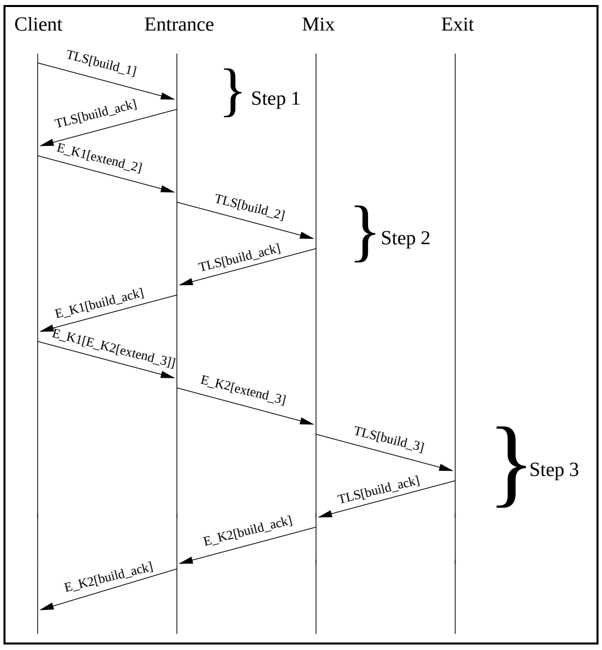

Figure 3: A sequential packet diagram of Tor's circuit building process. In Step 1, the client chooses the first hop along the circuit. Step 2 shows the chosen entrance router forwarding the client's request to the chosen mix router. Step 3 shows the entrance and mix routers forwarding the final circuit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

approach capable of doing so *before* the client starts to transmit any payload data. Furthermore, we experimentally verify the effectiveness of our approach in Section 4.

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 information for each cell received: (1) its location on the current circuit's path (whether it is an entrance, 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 onion router. Once this attack has been carried out, it is possible to determine which paths containing a malicious router at the entrance and exit positions correspond to a particular onion 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 3 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 entrance router and the entrance router sends an acknowledgment. Next, the proxy sends another circuit building request to the entrance router to *extend* the circuit through a chosen mix router. The mix router acknowledges the new circuit by sending an acknowledgment back to the client via the entrance node. Finally, the proxy sends a circuit building request to the exit node, which is forwarded through the entrance and mix routers to the chosen exit server. Once the exit router's acknowledgment has been received through the mix and entrance 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 naive packet counting approach would not be sufficient, since not all cells sent from the onion proxy are fully forwarded through the circuit; thus, the number of cells received at each onion router along the circuit is different. This pattern is highly distinctive and provides a tight time bound, which we exploit in our circuit linking algorithm.

Our circuit linking algorithm works as follows: The entrance node verifies that the circuit request is originating from an onion 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 entrance node is the same as the previous hop of the exit node. Finally, in step 3, it is verified that

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

terface between the HTTP client and the onion proxy is made possible by the `tsocks` transparent SOCKS proxy library [30]. 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. The majority of this bandwidth was consumed while handling control packets during the circuit-building process. This has the effect of enabling the malicious nodes to accept the majority of the circuit-building requests and, therefore, exist on a high number of circuits. Therefore, all the malicious routers' behavior is aimed at maximizing the probability that it will be included on a circuit.

4.4 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, the percentage of the Tor paths that our path linking algorithm presented in Section 3.2 can correctly correlate is measured. Also, to demonstrate the effect of the attack on the anonymity of the system, the total network entropy is calculated.

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 onion router deployment, we conducted experiments by adding two (2/42) and four (4/44) malicious nodes. The malicious routers composed roughly 5% and 9% of the network. 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 onion routers. 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 onion routers, the attack compromised over 46% of the 13,568 paths. The results as percentages of compromised paths are given in Figure 4 and 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 a total of 12 incorrectly correlated paths (one false positive in the 3/63 experiment, three false positives in the 4/44 experiment, and eight false positive in the 6/66 experiments). This low number of false positives shows that our path linking algorithm is highly accurate.

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

	Number of Malicious Routers	
	2/42	4/44
Analytical	0.12%	0.63%
Experimental	8.90%	33.69%
Improvement	7,574%	5,214%

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

	Number of Malicious Routers	
	3/63	6/66
Analytical	0.15%	0.70%
Experimental	11.06%	46.46%
Improvement	7,079%	6,546%

In Tables 3 and 4, the experimental results are compared to the analytical expectation of the percentage of paths that can be compromised by controlling the entrance and exit nodes. The analytical expectation is based on a combinatorial model originally defined in Tor's design document [9] as $(\frac{m}{N})^2$, where $m > 1$ is the number of malicious nodes and N is the network size. However, this analytical model does not take into account the fact that an onion router may be used only once per circuit. Thus, a more precise expectation can be described by $(\frac{m}{N})(\frac{m-1}{N-1})$. The analytical expectation given in Tables 3 and 4 is based upon this assumption. This analysis assumes that each node has an equal probability of being selected to be on a path. The difference between the analytical expectations and the experimental results is clear, as the experiments demonstrate an improvement of between 52 and

75 times increase in the number of circuits compromised. This shows how effective the attack is in influencing Tor's routing mechanisms, since the nodes are clearly not chosen with an equal probability.

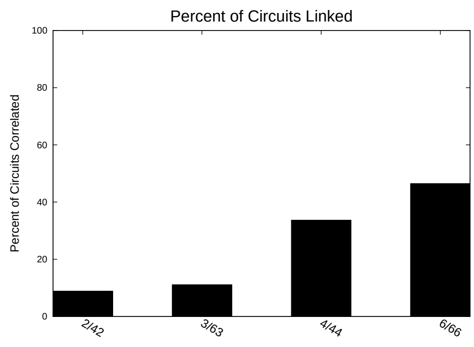

Figure 4: The percent of the network traffic that can be correlated versus the ratio of malicious nodes in each experiment.

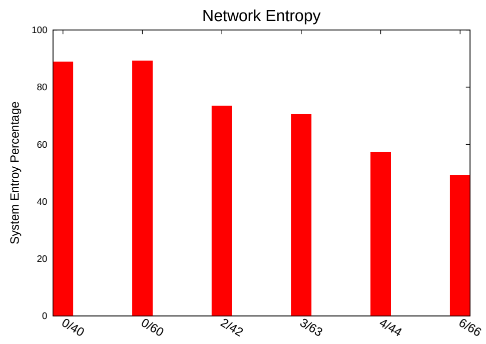

Figure 5: The network entropy is given as the ratio of malicious nodes varies.

We measure the network entropy, as defined in Section 2.4, by logging all of the paths used by our clients to traverse our isolated Tor network. It is not possible to measure the full entropy of the system in the real Tor network, since we do not control all of the clients using Tor at any given time. The entropy measurements for our experiments given in Figure 5 show that when no malicious nodes are present, the entropy of the system is approximately 0.9, which is close to the ideal entropy of 1.0. However, when 9% of the network is malicious, the entropy of the network drops to 0.49 in the 6/66 experiment and 0.57 in the 4/44 experiment. The decrease in network entropy when we added malicious nodes shows the global impact that our attack has on decreasing Tor's anonymity.

4.5 Cost of the Attack

Our attack is perfectly suited to a low-resource attacker, since it is not necessary to prove in any way that an onion router can, in fact, support the bandwidth that it advertises. Thus, a compromised dial-up or cable modem connection is sufficient to send false bandwidth and uptime advertisements. In addition, the results of our experiments show that the attack has a significant impact upon the anonymity of the system with only roughly 9% of the network compromised. In the 40 node network, the 9% malicious nodes were able to compromise one-third of the traffic and in the 60 network, the 9% malicious nodes were able to compromise almost one-half of the traffic. We expect, at worse, that the attack will scale $O(N)$, for a network of size N. However, there are many variables to consider, such as the effect of variable router quality. This analysis remains an open problem, due to the variable router quality, the increasing size of the real Tor network, and the Tor maintainer's request [7] 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 as opposed to only new Tor clients, selective path disruption, 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 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 onion 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. Most users will infer that the system is slow and not think there is foul play involved. A better design 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 mix or exit routers. Since circuit-formation failures are common in Tor, we suspect that this attack would largely go unnoticed. Currently, there is no automated mechanism in place to detect an attack such as ours.

Selective Path Disruption. If a malicious node does not exist at both the entrance and exit positions of a circuit, but at only one position, it can still cause the circuit to break simply by dropping all traffic along the circuit. This would cause the circuit to be rebuilt with a chance that the rebuilding process will create a path configuration in which both the entrance and exit nodes are malicious.

Displacing Honest Entry Guards. Recall that Tor uses special entry guard nodes to protect the entrance of a circuit. In order to be marked by the directory servers as a possible entry guard, an onion router must advertise an uptime and bandwidth greater than the median advertisements. 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 private Tor network, we successfully registered 20 routers all on the same IP address and different TCP port numbers.[2] Flooding the network with doctored router advertisements allows a non-global adversary to effectively have a “global” impact on Tor’s routing structure. Namely, this attack has the effect of increasing the median threshold for choosing entry guards, thereby, preventing benign nodes from being marked as possible entry guards. This attack could help guarantee that only malicious nodes compromise the possible entry guard portion of the router pool. Currently there is no automated way to detect an advertisement flooding attack in Tor.

[2]The trusted directory servers currently have no limits as to the number of routers that can be hosted on the same IP address. In theory, an attacker can register up to 2^{16} Tor routers to the same IP address.

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 de-anonymizing 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 actually set up 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 de-anonymize client requests to this target website. The critical idea is for the entrance router to watermark a client’s packets using some time-based watermarking technique, such as the technique used in Wang, *et al.* [31] or variants. The adversary’s malicious entrance routers could embed a unique watermark for each client-mix 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 mix 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, and we welcome future work directed toward addressing this challenge. We consider, however, methods for detecting non-global *low-resource* adversaries based upon a reputation system.

Although presented in the context of Tor, we explore a number of naive general defensive strategies first. These defenses revolve around verifying the information that will later be used as input to routing decisions and, in the case that the information cannot easily be verified (such as bandwidth), we investigate methods to mitigate the effects that false information have on route selection. In addition, we provide a detailed description of a robust method for detecting false routing information.

6.1 Naive Solutions

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 naive solutions to the problem of resource verification.

Verifying Uptime. A server's uptime could be tested by periodically sending a simple heartbeat message. 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 onion routers, also periodically verify the incoming bandwidth self-advertisements that are received from onion routers. The directory server could measure a router's bandwidth before publishing the routing advertisement and correct the bandwidth if it found that the router does not have sufficient bandwidth. The problem 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 performance load that would be placed upon the few centralized directory servers, this centralized bandwidth auditing approach would create a performance bottleneck in the system.

Distributed Bandwidth Verification. In order to detect false bandwidth advertisements, it may be tempting to augment the routing protocol to allow onion routers to proactively monitor each other. Anonymous auditing [28], where nodes anonymously attempt to verify other nodes' connectivity in order to detect collusion, has been proposed as a defense against routing attacks in decentralized structured overlays. A similar technique could be designed, not for collusion detection, but rather 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.

6.2 Our Solution: Local Observation Based Reputation System

Reputation systems as an incentive mechanism are a common technique in peer-to-peer overlays to detect "free-loaders," or peers that selfishly utilize resources while contributing little to the system [15, 19]. Our approach allows the clients to use first-hand performance observations from past circuits to detect when a router's performance deviates from its advertised resource claim. This is more difficult than in the case of most peer-to-peer overlays, since a client cannot immediately determine which router's performance is inconsistent with its advertisement. This is due to the multiple hop path structure that is imposed on every link through the system. However, if the client keeps track of the routers present on links that fail or under perform, over time it can statistically find those that are on a disproportionately high number of links that failed or deviated significantly from their advertised performance capability. As a response to such a detection, suspected malicious nodes can then be blacklisted for a period of time by the client.

Details. In order to define a reputation system for a multi-hop overlay, it is necessary to derive a method to reward servers that perform at, or above, their advertised bandwidth capability, and penalize those that perform below. The complexity of this task is compounded by the difficulty in determining which hop along the path contributed most to the poor performance. Our approach to constructing such a system is based upon an evaluation of the node's performance history and the update of a reputation parameter for each new observation of a particular node in the system. For one observation, the reputation value is adjusted by the reputation adjustment parameter, c, which is computed by evaluating the product of the probabilities of each node being chosen by the weighted path selection algorithm, scaled by the size of the network:

$$c = N \prod_{a_j \in P} \frac{a_j}{\sum_{i=1}^{N} b_i};$$

where $a_j \in P$ is the bandwidth advertisement of each router in the set of routers that comprise the client's path through the network, and N is the total number of routers in the system. Note that $|P| = 3$ in the case of Tor circuits. Initially, each ith router is assigned a local reputation vector, $\vec{r_i}$, equal to its probability of being chosen by the client. Recall that the probability of a router being selected is based up its advertised bandwidth capability, as distributed by the centralized directory servers in Tor. Thus, each ith router's local reputation vector is initialized to $\vec{r_i} = b_i \,/\, \left(\sum_{j=1}^{N} b_j\right)$. Both positive and negative feedback must be used to adjust the reputation value of routers. If a path under-performs, all of the reputations of the routers in that path should be decreased. Alternatively, when a path performs as expected (or better), the reputations of all of routers included on the path should be increased. Based upon the entire path's performance, for each ith router in that path, the local reputation vector is updated according to $\vec{r_i} = \vec{r_i} \pm c$; where c is added to old reputation value to reward each router in a path that performs at or above the minimum of all bandwidth advertisements for that path. The reputation adjustment parameter is subtracted to penalize each router in a circuit that performs below the minimum bandwidth advertisement for that circuit. The minimal performance expectation is computed by $m = \min\{a_j \in P\}$, which is the lowest bandwidth advertisement for the routers in the path. The actual performance observation for a given path is simply the bandwidth, p. Thus, each element of the local reputation vector is updated according to the following rules:

$$\vec{r_i} = \begin{cases} \vec{r_i} + c, & \text{if } p \geq m \\ \vec{r_i} - c, & \text{otherwise} \end{cases}$$

Since it may be the case that a single poor performing node limits the performance of an entire path, the other nodes will not incur significant blame for the poor performance, since the reputation adjustment parameter, c, is always very small. In addition, the calculation of c ensures that nodes advertising high bandwidth resources will be rewarded/penalized more than those that make lower resource claims.

Our reputation strategy is based upon the desire to dynamically decrease the selection probability of underperforming and potentially malicious nodes. This strategy also has the effect of increasing the selection probability of routers that outperform their claims. As the number of a client's observations increase, the selection probabilities will adjust to favor nodes that repeatedly perform well, and discount the likelihood that potentially malicious under-performers will be selected upon future paths.

Distributed Reputation System. One limitation to the local reputation system proposed is that the system will potentially take a long time to converge upon the set of malicious routers. To increase the system's convergence time, we propose that each client aggregate and distribute its local reputation vector, $\vec{r}$. One concern that arises when distributing local observations is that it reveals the nodes that a particular client used in previous paths. This compromises the anonymity of a system such as Tor. To overcome this inherent limitation, the client creates a *pseudonym* to use when reporting its $\vec{r}$ vector to maintain its anonymity. The centralized directory server uses a reputation algorithm similar to Eigentrust [17] to compute a global reputation vector that is subsequently distributed to all clients. By aggregating the reputation information, the system's convergence time will increase. In order to prevent malicious clients from influencing the global reputation system, the directory server assigns weights to each pseudonym's vector based upon how much they deviate from the mean of all vectors for each node in the system. A reasonable weighting scheme would roughly conform to a normal distribution, where the closer a vector's reputation is to the mean, the greater the weight.

6.3 Preventing Sybil Attacks

In order for any distributed reputation system to be effective, it is necessary to address the Sybil attack [10]. Currently, the directory server publishs routers without any limit on the number of advertisements from a single IP address or verification of the information in the advertisements. One of the proposed extensions to our attack is to flood the directory servers with enough false routing advertisements to displace all of the benign candidate entry guards with malicious ones. Since routers are marked by the directory servers as suitable entry guards if they advertise more bandwidth and uptime than the median of all onion routers, it is possible to do this if enough false routers are advertised to raise the median values beyond the values reported by benign routers. To help mitigate this kind of attack, the directory servers should attempt to prevent router advertisement flooding by limiting the number of routers introduced. Some strategies to implement this limiting might be to place restrictions such that only one onion router can by located on a single IP address or subnet. This would force the attacker to have access to a large number of IP addresses which are located in different subnets. A similar idea of enforcing location diversity in routing was proposed in Feamster and Dingledine [11]. This requirement does not prevent router flooding, but increases the resources required to perform such an attack.

The possibility that a router advertisement flooding attack can displace entry guards also raises the question: Is Tor's current use of entry guards wise? It might be better to require some thresholds, such as two days of uptime and 300KB per second of bandwidth, instead of using median values. This would make it impossible to displace existing routers that meet these resource requirements with a flood of malicious routers.

7 Previous Attacks Against Tor

There have been three previously published attacks against parts of the Tor protocol. Murdoch and Danezis [21] 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. An altered client that created circuits of length one would inject a large amount of traffic in timed bursts. If the same onion router used by the altered client was also used by another client, the victim's traffic would exhibit a delayed pattern matching the delay caused by the large amount of traffic injected by the altered client. This attack exploited the fact that Tor is a low-latency mix that does not explicitly delay traffic, and that multiple circuits routed through that same router interfere with each other. This attack also had the limitation that it could only trace the circuit back to the entrance router, but could not trace it back to the client.

Øverlier and Syverson [22] presented an attack that located hidden servers inside the Tor network. They deployed a single altered onion proxy, which also functioned as an onion router. The client continually builds a path of length one to contact the hidden service in the hope that it will also be included in the circuit that the hidden service uses to respond to the client's request. When this occurs, the attacker can use timing information to associate the two circuits, and thereby locate the hidden service. The predecessor attack presented in Wright, *et al.* [32] was used to statistically identify the hidden service as the server that appeared on these linked paths most often. When this research was conducted, Tor did not implement entry guards. However, they were added after these results showed that entry guards would help conceal the location of hidden services. The attack did not attempt to compromise the anonymity of a general purpose circuit, only those circuits used to connect to hidden services.

Another attack on Tor's hidden services is proposed by Murdoch [20]. The key to this attack is the observation that when a server is idle, its CPU runs at a cooler temperature. Since there is a distinct relationship between a CPU's temperature and its associated clock skew, the attack exploits the ability to recognize minute clock deviations that are consistent with a high load. These clock skews can be detected by analyzing the TCP timestamps on packets received from the target server [18] and compared to an induced load pattern. If a correlation is detected, then the location of the hidden service has been compromised. It is suggested that this technique may be used to construct covert channels and even allow an attacker to ascertain the geographical location of a server.

Our attack is the first that presents techniques to subvert path selection algorithms in a low-latency privacy enhancing system. We show that an attacker can infiltrate the Tor network and can fully compromise the anonymity of a large percentage of users. Our only assumption is that the attacker has control of a small number of low-bandwidth computers. We do not assume that the attacker controls the website or other service that the victim is visiting as in Murdoch and Danezis [21].

8 Conclusion

Despite the fact that it is generally acknowledged that end-to-end traffic analysis attacks are possible to launch against mix-network anonymity systems, there has been little research focused toward developing preventative measures. This is largely due to the assumption that these types of attacks require a prohibitively large amount of resources and, therefore, are not practical for large networks. However, we introduce a low-resource end-to-end traffic analysis attack against mix-network anonymity systems that is highly practical. Our basic attack stems from the use of preferential routing mechanisms that provide low-latency, high-throughput performance suitable for interactive applications. However, preferential routing without sufficient resource verification is dangerous, as an attacker can compromise the anonymity of a large amount of the communication channels through the network.

To demonstrate the effectiveness of such a generic end-to-end traffic analysis attack on a representative and widely used privacy enhancing system, we implemented this attack against an isolated Tor deployment on the PlanetLab overlay testbed. In our experiments, we showed that Tor is vulnerable to attacks from non-global adversaries that control only a few high-resource nodes, or nodes that are *perceived* to be high-resource. Our end-to-end attack on the anonymity of new Tor clients exploits the fact that the system utilizes a preferential routing algorithm that attempts to optimize for performance. We extend this basic attack to low-resource attackers whereby false resource claims are reported to the centralized authorities and this bad information is propagated throughout the network. Since there is currently no mechanism for verifying false resource claims, the attack is able to compromise a high percentage of Tor paths.

We experimentally showed that after deploying a few high-bandwidth, high-uptime servers, an adversary can, with high probability, compromise the entrance and exit servers on a route for new clients. Having compromised these two servers, we then presented and experimentally validated a new method for linking paths, thereby compromising anonymity by binding together both parties in a flow. In our experiments conducted on our isolated Tor deployment consisting of 60 nodes, our attack was able to correlate over 46% of circuit-building requests through the entire network. This is a significant increase over the 0.70% analytical expectation assumed by many anonymity systems analysts. We believe a key contribution of this work is our observation that , in this partiular case, analytical models fail to reflect the full complexities of a real deployment. Tor's analytical model predicted that a relatively low number of paths could be compromised with an attack similar to ours. However, this expectation failed to fully account for the heterogeneity in the network, and as a consequence, our attack performed far above expectations.

Over the course of extending the attack to a low-resource adversary, we noticed several recurring themes in the design that enabled the attack to succeed. We provide several concrete solutions designed specifically to mitigate the low-resource variant of our attack on Tor. These solutions focus upon verifying all information in the system that can influence routing decisions by allowing clients to incorporate local observations in their path selection process. In addition, we believe that location diversity, in some form, can greatly reduce the ease at which an adversary can deploy a compromised network to perform this attack. Our hope is that these attacks motivate further research in the area of designing and implementing preferential routing algorithms in anonymous overlay systems that deliver a high level of performance without compromising the security of any aspect of the system.

References

[1] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. S. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation* (New York, NY, USA, 2002), ACM Press, pp. 299–314.

[2] CHAUM, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 4*, 2 (February 1981).

[3] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (July 2000), pp. 46–66.

[4] DANEZIS, G., AND CLAYTON, R. Route fingerprinting in anonymous communications. In *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P 2006), 2-4 October 2006, Cambridge, United Kingdom* (2006), pp. 69–72.

[5] DÍAZ, C., SEYS, S., CLAESSENS, J., AND PRENEEL, B. Towards measuring anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)* (April 2002), R. Dingledine and P. Syverson, Eds., Springer-Verlag, LNCS 2482.

[6] DIERKS, T. RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1, April 2006.

[7] DINGLEDINE, R. Personal communication.

[8] DINGLEDINE, R., AND MATHEWSON, N. Tor path specification. http://tor.eff.org/cvs/doc/path-spec.txt.

[9] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium* (August 2004).

[10] DOUCEUR, J. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)* (March 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)* (Washington, DC, USA, October 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] GUPTA, M., JUDGE, P., AND AMMAR, M. A reputation system for peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video* (New York, NY, USA, 2003), ACM Press, pp. 144–152.

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

[17] KAMVAR, S., SCHLOSSER, M., AND GARCIA-MOLINA, H. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of WWW2003* (2003), ACM.

[18] KOHNO, T., BROIDO, A., AND CLAFFY, K. C. Remote physical device fingerprinting. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 211–225.

[19] MARTI, S., AND GARCIA-MOLINA, H. Taxonomy of trust: categorizing p2p reputation systems. *Comput. Networks 50*, 4 (2006), 472–484.

[20] 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).

[21] 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.

[22] ØVERLIER, L., AND SYVERSON, P. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy* (May 2006), IEEE CS.

[23] PETERSON, L., MUIR, S., ROSCOE, T., AND KLINGAMAN, A. PlanetLab Architecture: An Overview. Tech. Rep. PDN–06–031, PlanetLab Consortium, May 2006.

[24] REITER, M., AND RUBIN, A. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security 1*, 1 (June 1998).

[25] 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).

[26] SERJANTOV, A., AND DANEZIS, G. Towards an information theoretic metric for anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)* (April 2002), Springer-Verlag, LNCS 2482.

[27] SHANNON, C. A Mathematical Theory of Communication. In *Bell System Technical Journal* (July, October 1948), vol. 27, pp. 379–656.

[28] SINGH, A., DRUSCHEL, P., AND WALLACH, D. S. Eclipse attacks on overlay networks: Threats and defenses. In *IEEE INFOCOM* (2006).

[29] 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.

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

[31] 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.

[32] WRIGHT, M., ADLER, M., LEVINE, B. N., AND SHIELDS, C. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02* (February 2002), IEEE.

[33] ZHUANG, L., ZHOU, F., ZHAO, B. Y., AND ROWSTRON, A. Cashmere: Resilient anonymous routing. In *Proc. of NSDI* (Boston, MA, May 2005), ACM/USENIX.

A Non-Entrance Router Selection Algorithm

Algorithm 1: Non-Entrance Router Selection

```
Input: A list of all known onion routers, router_list
Output: A pseudo-randomly chosen router, weighted toward the
        routers advertising the highest bandwidth
$B \leftarrow 0$
$T \leftarrow 0$
$C \leftarrow 0$
$i \leftarrow 0$
$router\_bandwidth \leftarrow 0$
$bandwidth\_list \leftarrow \emptyset$
foreach router $r \in router\_list$ do
    $router\_bandwidth \leftarrow$ get_router_advertised_bandwidth($r$)
    $B \leftarrow B + router\_bandwidth$
    $bandwidth\_list \leftarrow bandwidth\_list \cup router\_bandwidth$
end
$C \leftarrow$ random_int($1, B$)
while $T < C$ do
    $T \leftarrow T + bandwidth\_list[i]$
    $i \leftarrow i + 1$
end
return $router\_list[i]$
```

At the lowest level, the algorithm computes B, the total amount of bandwidth from all known onion routers and creates a list of each router's bandwidth advertisement. It then chooses a pseudo-random number C between 1 and B. Each onion router from the router list is selected and its bandwidth is added to a variable T; if T is greater than C, then the most recently added onion router is included in the circuit, if not already included. If T is less than C, more onion routers are selected and their bandwidth is added to the variable T, until T is greater than C. Thus, this algorithm assigns a weight to each onion router from a probability distribution defined by the magnitude of each router's bandwidth advertisement. Most significantly, the more bandwidth a particular router advertises, the greater the probability that the router is to be chosen. This fact is fundamentally important in the implementation of our attack.