Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor

Tariq Elahi[†], Kevin Bauer[†], Mashael AlSabah[†], Roger Dingledine[‡], Ian Goldberg[†]
[†]University of Waterloo [‡]The Tor Project, Inc.
[†]{mtelahi,k4bauer,malsabah,iang}@cs.uwaterloo.ca [‡]arma@torproject.org

ABSTRACT

Tor is the most popular low-latency anonymity network currently available, protecting the privacy of hundreds of thousands of people every day. To ensure a high level of security against certain attacks, Tor currently utilizes special nodes called *entry guards* as each client's long-term entry point into the anonymity network. While the use of entry guards provides clear and well-studied security benefits, it is unclear how well the current entry guard design achieves its security goals in practice.

We design and implement *Changing of the Guards* (COGS), a simulation-based research framework to study Tor's entry guard design. Using COGS, we empirically demonstrate that natural, short-term entry guard churn and explicit time-based entry guard rotation contribute to clients using more entry guards than they should, and thus increase the likelihood of profiling attacks. This significantly degrades Tor clients' anonymity. To understand the security and performance implications of current and alternative entry guard selection algorithms, we simulate tens of thousands of Tor clients using COGS based on Tor's entry guard selection and rotation algorithms, with real entry guard data collected over the course of eight months from the live Tor network.

1. INTRODUCTION

Tor [9] is the most widely used volunteer-resourced anonymous communication network. It is designed to provide communicating parties with anonymity from their communication partners as well as from passive third parties observing the network. This is done by distributing trust over a series of Tor routers, which the network clients select to build paths to their Internet destinations.

When the Tor network was first launched in 2003, clients selected routers uniformly at random. However, for load balancing reasons, the router selection algorithm was changed in 2004 so that clients weight their selection by the amount of bandwidth that routers offer to the network; a router that offers more bandwidth to the network is selected more often by clients.

Another key change to the original router selection algorithm in Tor is the use of *entry guards*. The concept of entry guards emerged as a solution to safeguard against a variety of threats to end-user anonymity [5, 13, 18]. Guards were adopted into Tor with specific parameters that seemed likely to provide acceptable security and performance characteristics for the network and end users. Those parameters include the number of entry guards that a client begins with, and the amount of time a client can use his/her entry guard before switching (rotating) to new entry guards.

Context and motivation. Recently, there has been renewed interest in reevaluating these fixed parameters in combination with other network conditions, such as churn and load balancing, to more carefully determine the security that entry guards provide to users. Dingledine [8] formalizes the open issues related to Tor's entry guard design, which are paraphrased below:

- Quantify the vulnerability due to natural guard churn, which is the added client compromise due to guard nodes going offline.
- Quantify the client compromise rates at different amounts of adversarial bandwidth in the network.
- Quantify the vulnerability due to guard rotation and compare with natural churn. Which of these is the dominant contributor to client compromise? Also, how does varying the rotation periods affect the compromise rates?
- Quantify the compromise effects of different numbers of guards.

While analysis [13] provides evidence of security benefits and there is a consensus within the Tor community that entry guards provide performance and load balancing benefits, there is yet no empirical evidence of the effects and limitations inherent in their design and in their implementation.

Understanding and improving entry guards. To gauge the security and performance impact of entry guards in Tor and to provide direct answers to the questions above, we conduct an empirical analysis of Tor's entry guard selection and rotation algorithms by constructing a simulation framework called *Changing of the Guards* (COGS).

Contributions. This paper offers the following contributions to the field of anonymous communications:

- We present COGS, our simulation framework that is designed to provide quantitative data about guard design choices.
- With COGS, we conduct an empirical characterization of entry guards fueled by real data on Tor routers captured by the Tor Metrics Portal. In particular, we analyze natural churn, entry guard rotation, the number of entry guards chosen, and other parameters in terms of their effects on security and performance through large-scale simulation of Tor's current entry guard selection and rotation algorithms.
- We identify and investigate the trade-offs between the variables above from the perspective of security and performance impact.
- We present direct answers to open research questions posed by Dingledine with discussion on future guard design research.

Our results indicate that Tor's guard flag allocation process improves overall guard stability, that guard rotation is a major contributor of client compromise yet is self-limiting, and that for certain client/adversarial models using more guards provides far superior security than possible under Tor's current defaults.

Roadmap. The remainder of this paper is organized as follows: Section 2 provides the reader with a comprehensive overview of Tor's design, including the guard selection and flagging algorithms.

We present COGS in Section 3, and we provide our answers to the open questions discussed above in Section 4. We discuss some open issues in Section 5, and in Section 6, we outline related work. Finally, we present our future work in Section 7 and conclude in Section 8.

2. BACKGROUND

In this section, we present a detailed overview of Tor's design and system architecture.

2.1 Tor Overview

Tor provides a means to communicate over the Internet anonymously. A Tor client can remain anonymous from Internet servers, and the parties in communication can remain unlinked from each other from the perspective of an observer. We shall now discuss the mechanics of how this is achieved.

The Tor network is composed of volunteer-operated nodes called *Onion Routers* (ORs), also known as *relays* or *nodes*. These ORs provide network connectivity and bandwidth capacity for end-user traffic. Anyone may operate a relay and indeed a strength of Tor is the diversity and number of its network nodes. When an OR joins the network it announces its details, such as its network address/port, its donated bandwidth capacity, and its *exit policy*—stating to what Internet addresses and ports outside of the Tor network this relay is willing to send traffic—to the (distributed) *directory authority*. The OR will then be listed on the global list of relays and be a candidate for routing end user traffic.

An end user downloads the Tor client, also known as an *Onion Proxy* (OP), which on start up downloads the *consensus* document of all running relays as well as relay *descriptors* from the directory authority (or one of its mirrors). These documents contain the details of each relay that the OP uses to route traffic through the network. In order to protect clients against route bridging and fingerprinting attacks [7], these documents are updated hourly so as to provide a current and consistent picture of the network to all clients. Consensus documents are published precisely once per hour and descriptors are updated in real time as their contents change.

The directory authority also provides metadata in the consensus document that helps the OP route traffic more intelligently. In particular, the OP uses the consensus in the process of constructing a *circuit*—a path through the Tor network. By default, circuits consist of three ORs selected by the OP. We next describe the process of router selection that is performed by OPs.

Router selection. In the default setting, the OP selects ORs from a distribution that favours higher-bandwidth relays but also allows low-bandwidth relays to be utilized to some extent. The three ORs in the circuit are termed the *entry*, *middle*, and *exit* ORs. The OP communicates directly with the entry OR, the entry communicates with the middle OR, and the middle communicates with the exit OR. Finally, the exit OR communicates directly with the destination Internet server.

Although the number of circuits constructed is governed by immediate and anticipated need, a general rule is that each circuit is used for about ten minutes before the Tor client will begin using a fresh circuit.

The OP constructs the circuit as follows. The OP first picks a suitable exit relay—suitability being a function of the relay's configuration as an exit relay (which is communicated to clients with the `Exit` flag in the consensus document) and its exit policy. Next, the OP picks the entry OR while ensuring that all the relays have distinct /16 IP addresses and relay *families*.[1] (We provide more details on the constraints placed on entry selection in Section 2.2.) The middle node is then picked in a similar fashion.

Finally, the OP constructs the circuit using the three ORs in an incremental and telescoping manner. The OP negotiates cryptographic material with the entry OR and once an encrypted channel is established between them it asks the entry OR to *extend* the circuit towards the middle OR. The OP then negotiates cryptographic material with the middle OR—communicating through the entry OR—to establish an encrypted channel between them. The middle OR is then asked to extend the circuit to the exit OR and the process is repeated to establish a secure channel between the OP and the exit relay.

2.2 Entry Guard Relays

All Tor relays are donated and as such it is hard to know which ones can be trusted. It is easy, then, for the adversary to donate resources and participate in circuits. The danger is when the adversary controls both the entry and exit ORs on a single circuit. In this scenario the client address and destination address of the traffic are known to the adversary who, through tagging or traffic confirmation attacks [6,10,12], effectively deanonymizes the client. Following this previous work, we assume that these attacks are easy and accurate to carry out; however, the extent to which this assumption is true is beyond the scope of this paper.

Given enough time and the presence of adversarial ORs, the OP will eventually construct circuits that have malicious entry and exit ORs. Since Tor picks relays weighted according to bandwidth, a sufficiently resourceful adversary can deluge the network with high-bandwidth relays and increase the rate at which it can compromise circuits.

To mitigate this and related threats such as the predecessor attack [18] and locating hidden services [13], entry guards were introduced. They limit the impact an adversary can have on Tor's user base by effectively reducing the number of times each client selects her entry relays, thus slowing the rate of compromise and reach of the adversary.

Instead of picking a new entry every time a circuit is constructed, the OP maintains a *guard list* of a handful of pre-selected entry relays. When the Tor client constructs this list, it selects an expiry time for each of the guards in the list uniformly at random from the range of 30–60 days; after that time, the guards will be dropped and repopulated, as described in detail below. When circuits are constructed, the entry relay to be used is selected uniformly at random from the client's guard list. The rest of the circuit building process remains the same. The effect of this change is that if no malicious guard relays have been picked, the user is uncompromisable by the adversary until she picks new guards. The disadvantage is that if a client does pick a malicious guard then she has a higher probability of being compromised for the next 30–60 days. It is debatable if it is better to be compromised with some probability all the time or to be either completely safe, or else compromised with higher probability. Øverlier and Syverson [13] provide analysis that the latter is preferable and hence the guard mechanism is embedded in the Tor client code.

Moreover, since entry guards have the potential of negatively affecting the performance of the Tor network and security of its users, they need to be carefully selected. The main mechanisms in place are the directory authority, which assigns guard status to relays, and the guard selection algorithm executed by the Tor client. We next

[1] Operators of multiple Tor relays can voluntarily mark all the ORs they control as being in a common family.

Algorithm 1: Tor's approach to retrying unavailable entry guards

```
Input: Current time $T$, last attempt at time $E_\ell$ to contact entry
       guard $E$, $E$ has been unreachable since time $E_u$
Output: Return true if we should try to contact $E$, false
        otherwise
$d \leftarrow T - E_u$
if $E_\ell < E_u$ then return true
else if $d <$ 6 hours then return $T > (E_\ell + 1$ hour$)$
else if $d <$ 3 days then return $T > (E_\ell + 4$ hours$)$
else if $d <$ 7 days then return $T > (E_\ell + 18$ hours$)$
else return $T > (E_\ell + 36$ hours$)$
```

explain how the guard flag is obtained by ORs and how the guard selection algorithm is carried out.

Guard flag. All ORs in the Tor network are monitored for availability and bandwidth capacity by the directory authority. Relays deemed to be stable[2] and providing bandwidth above a certain threshold (currently the median of all relay bandwidths, or 250 KB/s, whichever is smaller [15]) are selected to receive the *guard flag* in the consensus document; this flag marks a relay as eligible to be included in guard lists. This criterion promotes ORs that will most likely be around for a long time and provide a level of bandwidth that will not likely cause bottlenecks. However, we find that there is large variance in actual guard bandwidth and stability. At the time of our experiments there were, on average, 800 routers with the guard flag. An important tension to note is that if the criteria are too selective, then few guards will be available, forcing more traffic through fewer nodes, at a cost to both network utilization and security. At the same time, if the criteria are too lenient, then less stable guards are likely to churn more often, leading to larger client guard lists, and an increased likelihood of selecting a malicious guard. This paper investigates this balance in detail.

Guard selection algorithm. Each client ensures that the number of guards—both online and offline—in its guard list is at least the default number at all times. If a guard goes offline, either temporarily or permanently, and there are fewer than two *online* guards in the guard list, a new entry guard is picked, but each previous guard is retried periodically, with an increasing back-off period[3], according to Algorithm 1. In addition, each of the relays in a client's guard list expires in 30–60 days as a *guard rotation event* occurs. The algorithm for picking a guard, in either scenario, is as follows:

- Read the consensus to find the set of relays with the guard flag set.
- Exclude guards already in the client's guard list, if any.
- Exclude guards in the same /16 IP block or family as any of the guards in the client's guard list.
- Select a guard at random from the remaining list of relays, weighted by the relays' *adjusted bandwidths* (see below).
- Assign a random expiration time 30–60 days hence.
- Repeat until the guard list contains the required number of guard relays.

The adjusted bandwidths used as weights in the above algorithm are based on values reported in the consensus for each relay, further adjusted by *utility weights*. Since Tor's bandwidth capacity is at a premium, and exit bandwidth capacity more specifically, this weighting mechanism is in place to make the most of these resources, so that the network as a whole does not suffer from overly poor performance. These weights are a function of the total bandwidth of each relay type, the total network bandwidth, and the relative bandwidths and relay flags of individual relays. As the bandwidth and relay composition of the network changes, the bandwidth weights of individual relays also change. Note that the weighted consensus bandwidths are scalars *without units*; it is best to think of them as "points," where relays with more points are more likely to be chosen in circuits. They are not actual bandwidth measurements, and so it becomes difficult to translate this metric to real-world client experiences.

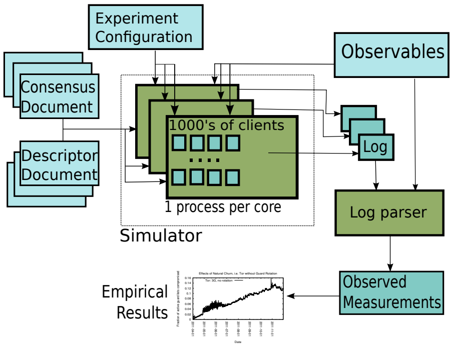

Figure 1: COGS framework

In general, exit bandwidth is protected such that relays with the `Exit` flag are chosen in the exit position more than in other roles. In particular, guards that are also exits will find themselves used more often as exits and less often as guards. This will have implications we will discuss later on.

Threat model. Tor provides anonymity properties against an adversary that has a limited visibility of the network. The adversary may operate malicious relays in the network and attain guard and exit flags by meeting the thresholds set out by the Tor specification. The goal of the adversary is to have relays under its control selected as the guard and exit relays on the same circuit, thus compromising the Tor user. The adversary does not have unlimited bandwidth and we count any relays it compromises as its own.

Our investigation of guards is concerned with the choices for parameters made by the Tor community. These parameters are the guard rotation duration, which at present is set to a uniformly random time between 30 and 60 days, and the number of guards, which at present defaults to three.

3. COGS FRAMEWORK

The design of the COGS framework was guided by Tor's guard path selection design, its governing parameters, the historical data sets available, and the research questions that we would like to answer. The design is extensible in that future research questions pertaining to guard and path selection can also be investigated using the same framework with minimal effort.

The framework encompasses a) researcher-defined observables or run-time measurements, b) the data sets available from the Tor Metrics Portal [16], c) a mass Tor client simulator with hooks into the internal running state of thousands of clients, d) configuration files that instrument the simulator for each experiment, and e) log parsers for data aggregation and statistics. Figure 1 provides a graphical representation of the framework. We will describe each in turn next.

[2] Not to be confused with the Stable flag from the consensus document given to relays with above-median mean-time-between-failures.

[3] While we do not analyze the effects of changing the back-off periods—currently believed to be orthogonal to Tor's guard design—COGS provides us the ability to do so in the future.

Observables. In order to drive the analysis and produce justifiable answers to the questions posed earlier, we define the following *observables*—metrics, attributes and effects that we want to measure. It is possible to introduce more observables for further research, some of which are outlined in Section 7.

From the historical consensus and descriptor documents we pick observables that will shed light into the behaviour of guards. We focus on client compromise and how it is affected by natural churn and the operational parameters chosen by the Tor community.

The pattern of up time and down time for each relay provides insight into its *stability*. Using the consensus history we measure the consecutive down times of each relay; the same is done for up times. From this we calculate the mean time to recover between two runs of up times as well as the mean time between failure between two runs of down times. Statistical analysis provides the average case for the general population of relays as well as that of guards.

To measure the impact of guard selection we also record the *number of guards that observe each client during the simulation*. This is a useful indicator since it establishes the high water mark of potential compromise for each client. Even though each guard may have only been an active guard to a client for a short period of time, it is not safe to assume that the short period afforded limited impact on the client's privacy, since that short period may have been very sensitive in nature.

Additionally, we measure the number of clients at each consensus for whom at least one malicious guard is in the *active guard list*; we term this event *guard list compromise*. The active guard list is the first N online relays in the client's guard list ordered by age, where N is the number of entry guards being utilized by the client. This provides a view from the adversary's perspective of how many clients it could potentially compromise at any given time. Whereas Tor will always maintain a minimum of two online guards, we experiment with active guard lists that at times shrink to one in Section 4.

Finally, in order to evaluate the effect on performance of reduced active guard lists that may occur due to changes to Tor's default behaviour, we measure the occurrences of active guard lists whose *average bandwidth falls below a certain threshold*. The number of active guards is not as important here as the average of their bandwidths, since this can directly influence the client's expected performance. We measure the average active guard list bandwidth as an indicator of the end user's experience and not as an expectation of the performance of any particular circuit. Recall also, from Section 2.2, that the weighted consensus bandwidths do not represent absolute bandwidths; nonetheless, we can meaningfully compare the schemes against each other to find the relative merits of each.

Data sets. The Tor Metrics Portal [16] provides hourly snapshots of publicly downloadable Tor relay descriptors and actual published consensus documents from mid-2007 to the present. This data offers a glimpse into the state of the Tor network over the past several years in terms of the total number of relays, their flags, and their bandwidths. In addition, the presence (or absence) of any particular relays enables us to track and analyze relay stability over time.

Configuration files of run-time options. We can change the behaviour of Tor clients, the adversary's attributes, and the network characteristics by passing parameters at run time through configuration files. Many experiments can be run simultaneously and independently—contingent on compute and storage resources—to provide insights into the behaviour of stock Tor and the many interesting variations open research questions introduce. This mechanism allows us to attain answers in an efficient and reproducible manner. We discuss our parameter choices below in more detail.

Tor path selection simulator. Using the publicly available data sets and our selected observables, we constructed a Tor path selection simulator that selects guard relays and generates paths for a large number of simulated Tor clients. The simulator takes two pieces of data and a configuration file as input:

1. *Consensus documents*: The simulator reads unmodified consensus documents, one at a time, over the course of the time period desired. The consensus provides information such as each relay's bandwidth weighting and its flags.
2. *Relay descriptors*: The simulator also reads in relay descriptors that correspond to each relay listed in a particular consensus to allow correct Tor client behaviour.
3. *Run-time options*: The simulator takes run-time parameters to introduce an adversary (if one is modeled), augment the behaviour of clients (if required), choose the number of clients to be simulated, and produce logs depending on the observables.

In order to ensure the highest possible level of fidelity to Tor's design, our simulator is based on Tor's original source code (version 0.2.2.33). For each consensus period, the simulated clients select or update their guard lists, following all of the Tor rules for guard replacement as described in Section 2.2.

Our simulator allows us to control the guard rotation mechanism built in to Tor to test the effects of various guard rotation durations (or lack of) on client compromise and also allows us to investigate the effects of client guard list size.

The granularity of our simulations is one hour, which corresponds to the granularity of the consensus documents. Every consensus lists the relays that were available at the time; they are loaded into the memory of our simulator, which then proceeds to select guard relays according to Tor's procedure for every client. These guards are written to a log file for later processing. Each consensus is fed into the simulator as a means to walk through time and produce guard selection scenarios using parameter settings provided by us to simulate different network characteristics such as the number of guards, guard rotation period, and others. Where consensuses are missing from the Tor Metrics dataset, the simulator skips that hour of history but all time-sensitive rules and operations are followed and are reflected in the simulation results.

We can simulate an adversary with a fixed budget of relay bandwidth by injecting it into the list of routers in each consensus period. The adversary is modeled by the amount of bandwidth it owns and the number of nodes it controls.

We also instrument the Tor client code to log client state to disk for all observables we are interested in. We refrained from logging all state changes due to storage constraint considerations.

We have made COGS available as open-source software and it is available from http://crysp.uwaterloo.ca/software/.

Simulation setup and parameter choices. Our simulations were run on multi-core servers to take advantage of parallelism in the experiments. Each simulation run introduced 80,000 clients.[4]

It is not yet clear how to best model the client behaviour as there is yet no consensus within the Tor community on real-world client behaviour. Indeed, this is a research problem in itself and out of the scope of this work. Therefore, we model the user base size as constant with no new clients joining the network, since our simulations focus on long-term effects that are not sensitive to user churn. For

[4] This sample size was chosen by conducting preliminary experiments of increasing sizes, and finding the point at which the result distributions stabilize, according to the Kolmogorov-Smirnov distance.

simplicity, the clients in our simulation are always online, which is a worst-case scenario since real-world clients likely do not use Tor continuously.

We choose the duration of our simulation by providing the starting and ending epoch times. We chose Apr 2011–Nov 2011 as our target time slice since it has relatively stable bandwidth characteristics and a consistent consensus version number.

COGS allows the injection of malicious routers into the network at run time through a configuration parameter.[5] We have chosen to introduce the malicious relay one consensus period, i.e. one hour, after the simulation has begun and all clients already have honest guards in their lists. This simulates an adversary attacking Tor after clients have already started using it and also establishes more conservative compromise rates—effectively a lower bound. For our simulations we assigned our malicious relay the guard flag only since also having the exit flag reduces the probability of a router being picked as a guard relay and would confound our results. Note that the choice to operate an exit node is with the relay operator and is not controlled by any authority.

The bandwidth assigned to this relay, approximately in the top 20% of guards, is incorporated into the network using the same rules and bandwidth weightings as the normal routers. Using the results from Murdoch [11], we only introduce one malicious relay because Tor's guard selection algorithm chooses guards in proportion to their bandwidth; this means that an adversary operating one high-bandwidth relay is equivalent to one operating many low-bandwidth relays as long as the total bandwidths are the same. Since we consider the adversary to be intelligent and capable of leveraging any and every advantage, we consider a client to be compromised if even one malicious guard exists in her active guard list.

It should be noted that while we initially set malicious bandwidth as a proportion of the total bandwidth, this proportion changes over time along with the total network bandwidth. We reason that keeping this value constant does not harm the experiments since i) a real adversary would not measure bandwidths on the entire network to keep malicious bandwidth proportions constant and ii) the bandwidth variance is small in our selected time period.

Log parser and data visualization The log parsers—there are several variants depending on the observables—extract the data we are most interested in and compile it into a format that can then be fed into data visualization programs. The raw data can be processed by different parsers to gain insight into the various aspects of guard design.

Our existing parsers process the raw logs to provide data on client compromise rates, total guard exposure over the experiment run and expected client performance.

While COGS is rooted in guard analysis, it can also be used to simulate other Tor-related phenomena that do not involve actual network traffic. Examples include the analysis of client circuit diversity, the effects of introducing exit guards, and assessing whole-network effects of heterogeneous client configurations.

4. MEASUREMENTS AND EVALUATION

We now use COGS to collect the empirical data that will be used to answer the four open research questions introduced earlier. The main aim is to understand the effects of various guard design choices on client compromise rates. We measure the frequency with which a client picks new guards, since the more often guards are picked the more often a malicious relay has the chance to be placed in the client's guard list. The two main influences on the frequency of guard selection—other than a new client joining the network—are *natural churn* and *guard rotation*. We measure and evaluate characteristics of each in order to better understand the threats to client privacy.

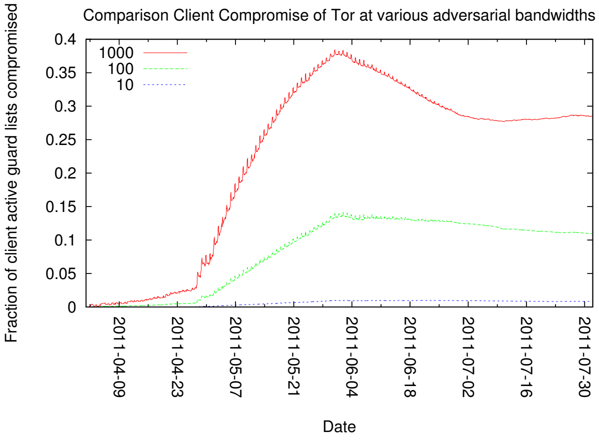

Figure 2: Client compromise rates at various adversarial bandwidths.

Table 1: Up and Down times in hours of guard relays for Apr–Nov 2011

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Guard Down	1	1	3	42.17	11	4978
Guard Up	1	7	20	156.7	127	3829
All Down	1	3	10	45	20	5454
All Up	1	1	4	19.82	11	3829

Adversarial bandwidth and compromise rates. We base the subsequent analysis on the assumption that malicious bandwidth is directly related to client compromise rates, albeit in complex ways. As the adversary increases their bandwidth contribution they are able to compromise more clients. This is by design, as the Tor guard selection algorithm favors relays with higher bandwidth. We provide confirmation of this assumption in Figure 2 which shows that as the malicious bandwidth increases the compromise rates also increase.

Since bandwidth is independent of the other variables under study, we keep the adversarial bandwidth constant for the rest of our experiments.

4.1 Natural Churn

To measure the effect of natural churn, we start by first analyzing the consensus data and establishing the pattern of churn (e.g., up and down times) for each relay over time. The subsequent statistical analysis provides the results in Table 1. Note that we allow for the effects of relays that had a high frequency of up/down events, and that only relays that were available April to November 2011 were included in the data set.

From the distance between the upper and lower plots in Figure 3 we find that guards are more stable compared to the general router population due to their relative longer up times and shorter down times.

Next, we measure the effect of natural churn on guard list compromise and present the results in Figure 4 as the lower plot. For this analysis we have removed the normal guard rotation mechanism in the Tor client to isolate the effects of natural churn. We note that natural churn occurs frequently and also has a large effect on

[5] While this paper investigates only one value of this parameter, it is simple to instantiate other behaviours through other values.

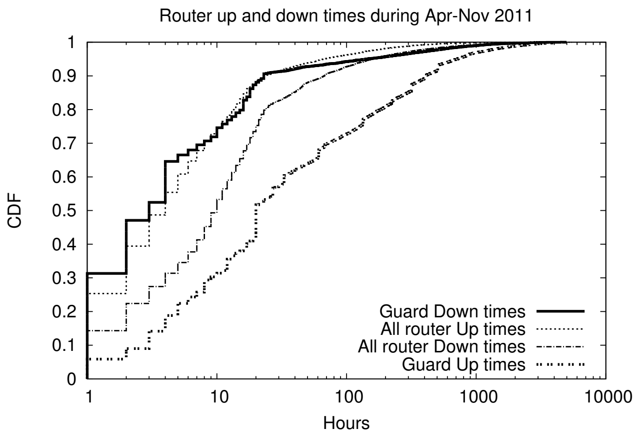

Figure 3: Router up and down times for all routers and for guards alone.

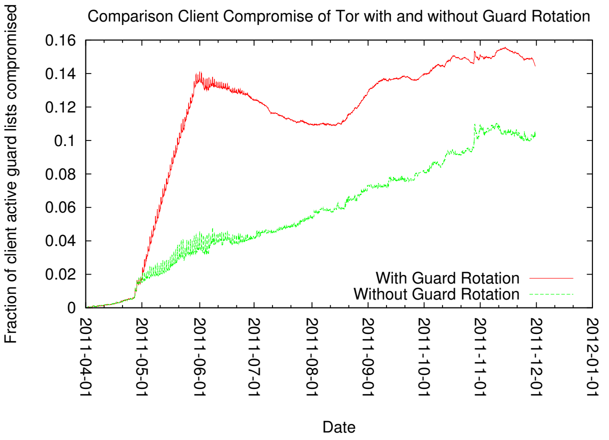

Figure 4: Effects of natural churn and guard rotation on compromised guard lists

the network as indicated by the large uptick in compromised guard lists over time. The sharp peaks and valleys between May 1, 2011 and June 30, 2011 are indicative of honest guards that go down briefly—during which time our malicious guard has an opportunity to move into the active guard list—and then return—which bumps the malicious guard out of the active list again. These characteristic short guard down times concur with both Table 1 and Figure 3.

From the upward trend of the plot we now know that natural churn has a real and lasting effect on client security and increases with time. Given enough time a long-lived adversary will appear in all clients' guard lists. This risk can be mitigated with periodic guard rotation, which is presented next.

4.2 Guard Rotation

The second factor to guard list compromise is the mechanism to rotate each client's guards after defined periods of time. By default a Tor client drops its guards that are between 30–60 days old in the guard list. There are two major reasons: to limit the number of clients a single well-resourced guard can service, and hence compromise, at any given time and to balance the load so that long-serving guards do not potentially end up bearing the load of more clients over time. A negative effect is that clients with all honest guards are exposed to potentially selecting a malicious guard upon rotation, thus ensuring that after enough time all clients will have been compromised at some point.

It is difficult to isolate the effects of guard rotation from those of natural churn under simulation with real data. We can, however, analyze the effects of guard rotation in closed form and also analyze the empirical results of the additional effect of guard rotation to natural churn in simulation.

During our target time slice of eight months, we expect that every client will rotate their guards at most as often as 30 days and at least as often as every 60 days. The maximum number of potentially unique guards that a client selects in those eight months is therefore 24, the minimum is 12, and the average is 17. This is the number of guard relays that can potentially compromise the client. Note that without guard rotation, the least number of guards per client would be three.

The upper plot in Figure 4 shows the additional effect that guard rotation has on client compromise rates. In the first 30 days we see a steady increase on both plots in compromise rates as only natural churn is in effect. Then between 30–60 days the guard rotation really begins to show its effects in the upper plot, peaking at the end of May after which point a steady state seems to have been reached, where the amount of new compromised active lists is offset with losses in compromised active lists. The downward and upward trends are due to our malicious relay's relative bandwidth fluctuating as the total network bandwidth fluctuates.

It is obvious that guard rotation increases the chances of active guard list compromise substantially. This implies that guard rotation has a larger effect on compromise than does natural churn alone. Although it is difficult to isolate the interplay of natural churn and guard rotation it is simple to see that guard rotation does have negative effects.

A key takeaway here is that the nature of guard rotation and natural churn are different, which explains the disparity between the plots. Guard rotation *replaces* a guard, while natural churn only provides a *backup* guard. If the client picks no malicious guards (as is the case initially), then with only natural churn in effect the malicious relay can only hope to be picked once a client's guard goes offline. However, it will *never* be at the top of the list and will be bumped out of the active list once the original guard returns. On the other hand, when guard rotation is used, every 30–60 days the malicious relay gets a chance to be picked as one of the first three guards thus cementing its place in the active guard list and thus allowing potential compromise whenever it is the selected guard relay.

Figure 5 shows the distribution of the number of clients that have been seen by various numbers of guards for Tor with and without guard rotation. Guard rotation greatly increases the visibility of each client on average to 19 guards. Recall that rotation causes at least 15 guards to see the client at minimum, so coupled with natural churn this effect is amplified. The effects of natural churn alone are small according to this metric: the mean increases to five versus the minimum three guards per client as indicated by the left plot.

As a counterpoint we observe that guard rotation does serve a beneficial purpose. As mentioned earlier, it reduces the likelihood that certain long-lived guards will accumulate a large set of clients and hence potentially compromise them. This self-limiting nature means that it is not desirable to remove guard rotation as a mechanism without a suitable alternative; we are actively exploring this area as ongoing work.

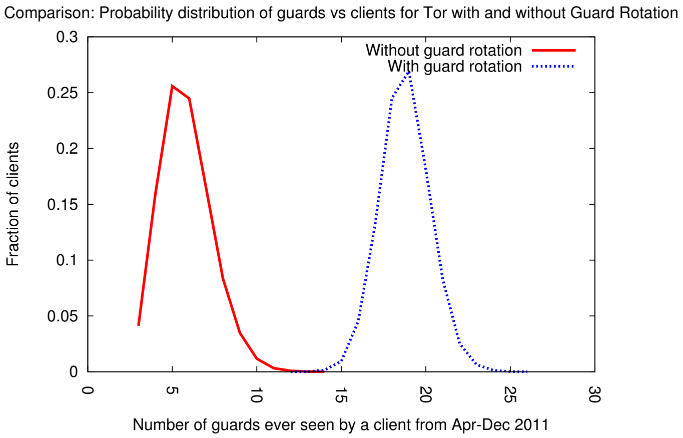

Figure 5: Comparison of natural churn and guard rotation effects

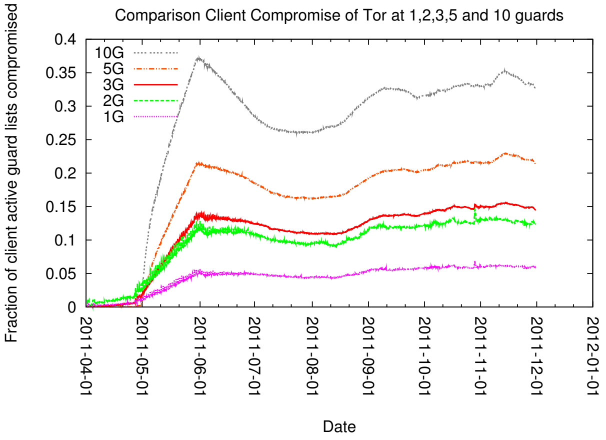

Figure 6: Client compromise rates at various client guard list sizes, with guard rotation.

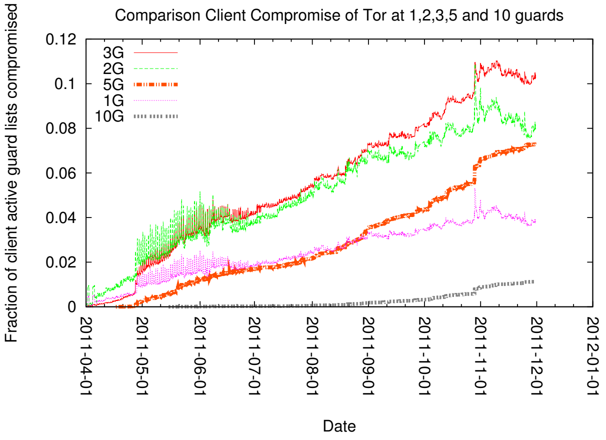

Figure 7: Client compromise rates at various client guard list sizes, without guard rotation.

4.3 Client guard list size

Finally, we investigate the effects of the size of the client's guard list and provide results and analysis for various values. We include results both with and without guard rotation enabled.

Recall that the client will only replace a guard if guard rotation dictates it (if in effect) or when there are fewer than two guards online from the client's guard list. Figure 6 shows client compromise rates with guard rotation when the size of the client's guard list is 1, 2, 3, 5 and 10 guards, where the 'G' stands for guards. From this we discover that increasing the size of the guard list increases the client compromise rates.

However, compare this to the results without guard rotation in Figure 7, where the absolute compromise rates are far lower but steadily increase over time and would eventually surpass the guard rotation plots. Also note that with guard rotation off, increasing the guard list size beyond 3 guards has the reverse effect of decreasing client compromise (plots for 5 and 10 guards). However, this effect does not last as we see the plot for 5 guards crossing over the 1 guard plot and would eventually cross over 2 and 3 guard plots as well. The same trend occurs for the 10 guard plot. The reason behind this is that initially the pool of possible guards is large and all are online; as guards fail, the client does not take any steps to replace them since the size of the guard list is still large enough and at least two of them are online. As the guards that failed are removed from the list, more guards are picked to maintain the overall size of the client's guard list. This last effect slowly erodes the advantage of starting off with a large pool of guards.

We now consider the number of guards seen over time for different values of starting guard list size. Figure 8 shows the effect of increasing guard list size on clients' guard exposure. It becomes apparent that increasing the guard list size increases the client's guard exposure.

Figure 9 provides results for when guard rotation is turned off. While the overall guard exposure is far less than when guard rotation is in effect, we see the same trend where larger starting guard list size equates to more guard exposure. We observe that as the client guard list size increases, the probability of more guards ever being added to the list decreases. This is particularly striking for the 10 guard plot, and also evident for the 5 guard plot. This is both an effect of relative guard stability and due to the condition that fewer than two guards be present before a new guard is added. Comparing both figures we see a more general trend that without guard rotation the value of guard exposure is close to the starting guard list size (more pronounced for higher values) whereas with guard rotation the values of guard exposure are many times larger.

4.4 Available bandwidth

Before we can make any conclusions we must look at the effects of these parameters on the average available bandwidth a client's guards provide it. Figure 10 shows the expected bandwidth for a client circuit. Results with and without guard rotation are nearly identical with negligible variations meaning that average performance is independent of guard rotation.

Recalling that higher-bandwidth guards are more likely to be selected for spots in a client's guard list, poor guard bandwidth availability happens when *all* of a client's active guards have low bandwidth. This occurs with decreasing probability as the number of active guards increases, and this is reflected in the dramatic decrease in the long left tail in Figure 10 as the number of guards increases from 1 to 3. Above 3, however, the improvements are less pronounced.

In this section, we measured and analyzed the effects of varying the parameters of Tor's guard design. Next we discuss the implications of these findings, and address Dingledine's open research questions.

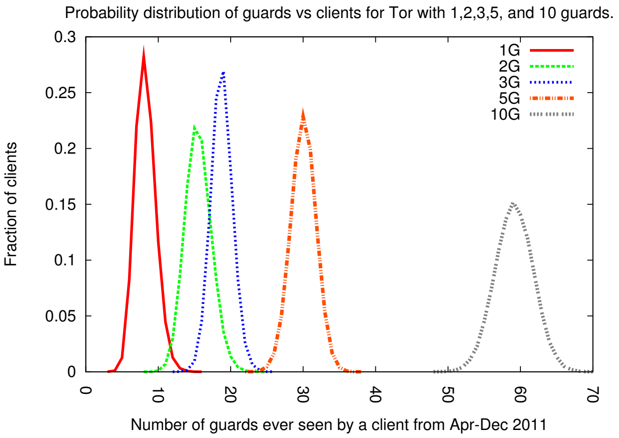

Figure 8: Client guard exposure with guard rotation at various guard list sizes.

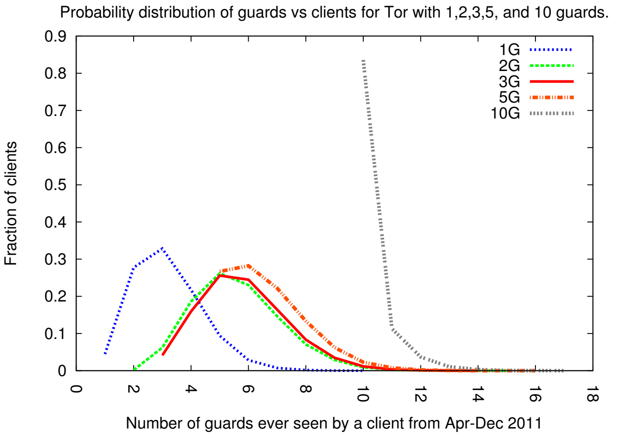

Figure 9: Client guard exposure without guard rotation at various guard list sizes.

5. DISCUSSION

Guard stability and selection. As guards are the first hop on circuits, all of Tor's functionality is contingent on their availability. We explored their stability and found that on the whole guards are quite stable. Compared to the general population of relays, guards are generally available for longer stretches of time and offline for shorter durations. This is a consequence of the guard flag distribution process governed by the directory authority and is as designed.

However, this process is not perfect, as we see that there are a large quantity of guards with a wide variety of stability characteristics that deviate from the intended entry guard design—recall Table 1 for the range of downtimes and uptimes for guards. We note that the incidence of active guard lists with low average bandwidth in general is not prevalent; note that the plots for 3–10 guards in Figure 10 do not have long tails to the left of the median as compared to the 1 guard plot—meaning occasions where the entire active list has low bandwidth are rarer—and that perhaps the guard flag could be more selective. Indeed, Table 2 provides statistics on the median guard bandwidth during our 8-month time slice; it shows that the maximum of median guard bandwidth is just 113, a level of active guard bandwidth which is surpassed by all clients under "3G" and above and only suffered by 5% of "1G" clients (Figure 10).

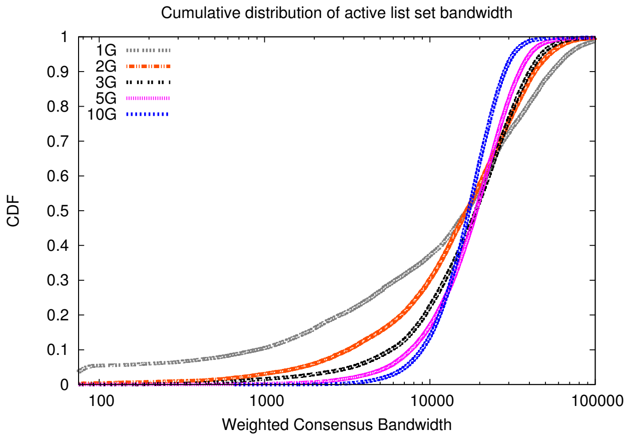

Figure 10: Client's expected circuit performance *with* guard rotation at various guard list sizes. Performance results *without* guard rotation are nearly identical.

Table 2: Median guard bandwidth from Apr–Nov 2011

Min	Median	Mean	Max
40	67	68.31	113

We reason that since low-bandwidth guard lists are rare, low-bandwidth guards are not depended upon by end users and so removing them from guard lists will not have a big impact from a performance perspective.

However, it can be argued that for the sake of load balancing these low-bandwidth relays provide relief whenever the end user chooses them from their guard list instead of one of their higher-bandwidth guards. These nodes may also provide added security through additional relay diversity. It is unclear at the moment if these nodes actually fulfill these desired effects, and this is an avenue of future investigation.

Natural churn and its effects on client compromise. In the lower plot in Figure 4, it is clear that natural churn provides an adversary increased opportunities to compromise guard lists. We also note that although there is some downward pressure due to returning honest guards, the trend is upwards over time. If not for guard rotation, after a sufficient length of time a malicious relay should be able to compromise all client lists. Recall that while guard rotation speeds up the adversary's accumulation of clients initially, it is self limiting as the rate of clients gained equals the rate of clients lost due to churn.

These trends indicate that guards are not dependable, but that this is an unavoidable consequence of the volunteer nature of the Tor network.

Furthermore, when reasoning about the impact of natural churn it is difficult to know beforehand when a guard is likely to return, if ever. It is due to this uncertainty that Tor has opted for such sensitive guard replacement policies, and sophisticated guard retry mechanisms.

Putting natural churn in perspective, we can reason that it is an artifact that cannot be removed from the network, and it has a large effect on the security and performance of the network. Therefore, the best policy may be to avoid situations that lead to churn in the

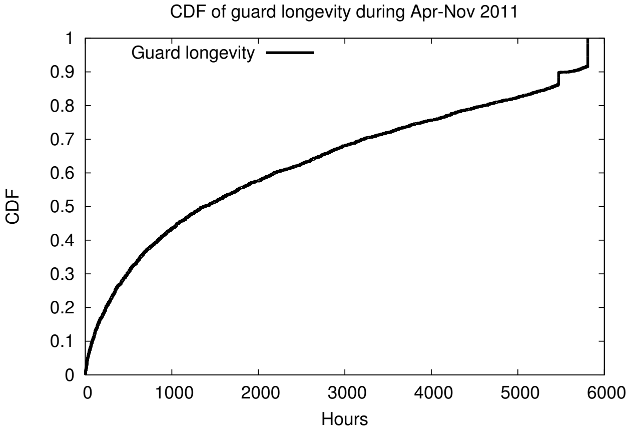

Figure 11: Guard longevity during Apr–Nov 2011.

first place by selecting guards more cautiously and mitigating the effects of churn when we do find one of our active guards offline.

Guard rotation. Long-lived relays will tend to accumulate clients over time, and malicious relays will remain online to take advantage of this effect. Rotating guards periodically mitigates that eventuality.

We note that guard rotation does, however, increase the chance of compromise. This can be seen by the sudden increase in May 2011—when guard rotation began to take effect in our experiment, 30 days after its beginning—of the plot in Figure 4. We also note that in Figure 5 the number of guards a client is serviced by, and can hence potentially be compromised by, is much larger when guard rotation is in effect. Furthermore, Figure 9 indicates that all schemes expose clients to fewer guards when guard rotation is not enabled. As mitigation of the above, we could increase the minimum and maximum durations of guard rotation from the current 30–60 days and see a reduction in both metrics, since the frequency of rotation events would decrease.

In order to reason about rotation durations and pick better ones we plot in Figure 11 the CDF of guard longevity for Apr–Nov 2011. We note that only about 9% of guards remained part of the Tor network for the entire 8-month duration of our experiments. Also, the distribution is skewed towards shorter-lived routers with the median at 1371 hours. The current rotation period is between 720–1440 hours which means that the majority of guards undergo guard rotation. Since most guards are not long lived and leave the network of their own accord, guard rotation occurring as frequently as it currently does is both unnecessary and undesirable. We would prefer to target only those guards that are truly longer lived and thus are the cause for our concern, and ignore those that simply do not exist long enough to be worried about. Unfortunately, there is not an upward inflection point, apart from the two small ones at the extreme end of the time slice, which would indicate that longer-lived guards stand apart from the others and thus can be dealt with using a more appropriate rotation duration.

Perhaps as an alternative for longer-lived, and potentially more-utilized guards, Tor ought to adjust their probabilities of being selected according to how long they have been part of the network, in addition to their bandwidths. The directory authorities can estimate the number of clients currently using each guard relay based on its historical weights and bandwidths, and adjust the weights in order to balance the load. In this manner, over time, each guard will effectively limit the load on itself as well as reduce the accumulation of clients.

Tor with One Guard. Intuitively, it seems that one guard ought to provide the best security but that perhaps performance would suffer. We revisit the results in Section 4 to evaluate this intuition. From a circuit compromise perspective, we see in Figure 6 that Tor with one guard offers the least likelihood of compromised guard lists. We also note that fewer guards participate in an end user's guard list in that case (Figure 8). However, from a performance perspective we note in Figure 10 that compared to Tor with three guards, Tor with one guard suffers from 60% worse performance 50% of the time but is better 50% of the time where it provides 25% more average guard list bandwidth. This can be explained with guard lists that have a combination of slow and fast guards, which causes the average to be lower than the fastest guard. In the case of Tor with only one guard, when a fast guard is selected, the client can expect to receive fast service, provided that the middle and exit nodes are not slow. It is important to note, however, that Tor with one guard is superior to the other schemes evaluated in Section 4 when the bandwidths are already at acceptable levels, whereas it provides far slower performance at the lower ends of the bandwidth spectrum.

Hence, the number of guards is a parameter that needs careful adjustment, since our present results suggest that too few may lead to performance degradation, while more can have unnecessary security implications. Indeed, it may be the case that there are hereto undiscovered security implications of using just one guard as well. COGS provides a way to investigate these questions and find solutions.

Answers to Research Questions. We now tie the results of our investigation to the questions posed in Section 1, and see how much progress has been made and what remains to be answered.

We found that adversarial bandwidth is directly related to client compromise rates and that this is an unavoidable effect of favouring higher bandwidth relays—recall that this is a design choice, for the sake of better performance. What is interesting is that, as seen in Figure 2, bandwidth and compromise rates are not linearly related. While more research is required to establish the exact relationship, it is clear that performance-enhancing measures have led to higher client compromise rates in this regard.

We found that users achieve greater security if they reduce the number of entry guards they use. They can further improve their security by eliminating or reducing the process of guard rotation. However, we also found that the security improvements through the reduction of guards and guard rotation come at the expense of performance degradation. We also found that natural churn, while inherent in the network and a source of compromise, works to amplify the compromise rate, but is not a dominating factor in the present Tor network.

We did not vary guard rotation periods in this paper, and this is an avenue of future work. We hope to identify an alternative mechanism that is independent of relay stability and network characteristics.

By putting all the parameters together we find in general that if a suitable alternative to guard rotation can be found and smaller guard lists used, then the security of Tor's users will increase significantly while the impact to performance for clients with slower-than-average guards will degrade only slightly.

This could be further mitigated by making the guard flag more selective and thus removing low-bandwidth guards, which would raise the average guard bandwidth for all clients. We identify this as an area for further analysis.

6. RELATED WORK

Entry guards were first proposed by Wright *et al.* [17] (there called "helper nodes") to mitigate the threat of the predecessor attack [18] in low-latency anonymity networks. In the predecessor attack, an adversary who deploys relays into the anonymity network can passively link possible senders with possible receivers. If clients choose their paths through the anonymity network by uniformly random selection, the predecessor attack predicts that an adversary that controls c out of n nodes has the expectation of successfully observing a given client after c/n rounds; the same adversary has a $(c-1)/(n-1)$ probability of observing the corresponding destination server and a $(c/n)((c-1)/(n-1))$ probability of linking the two. To eliminate the predecessor attack, Wright *et al.* propose that the first node in a path be fixed. Clients who have the misfortune of choosing a malicious entry node are guaranteed to have a compromised first hop, while all other clients are protected from this threat.

Entry guards for modern onion routing networks (like Tor) were proposed by Øverlier and Syverson [13]. Since Tor does not choose circuits with uniform selection over the available nodes (but instead, in proportion to each node's bandwidth capacity), the details of the analysis of the predecessor attack are more complicated. However, Øverlier and Syverson found that an adversary who artificially inflates his perceived bandwidth capacity will be selected more often and can launch a powerful predecessor attack. To mitigate this threat, they propose that Tor clients choose a small, fixed number of Tor relays to always use as entries into the anonymity network.

Extending Øverlier and Syverson's attack, Bauer *et al.* [4] showed that an adversary who controls a large number of nodes can launch a type of Sybil attack that has the effect of replacing all non-malicious entry guards with malicious ones (potentially all running on the same machine). The attack works by deploying enough malicious nodes that advertise high bandwidth and uptimes to effectively raise the criteria for the guard flag so that only malicious nodes can be used as entry guards. This attack was dangerously easy to launch, due to the fact that Tor's authoritative directories relied solely on self-reported (and potentially inflated) bandwidth and uptime claims. In part due to this attack, the directories now track each router's bandwidth and uptime [2, 14], and ensure that no one can launch too many malicious nodes from the same machine (or network) [3].

Borisov *et al.* [5] describe the effects of entry guards on the selective denial of service (DoS) attack. They argue that while the selective DoS attack will never be effective on a client that uses honest entry guards, the attack becomes more powerful when a client uses malicious entry guards. The authors also suggest that the choice of three entry guards results in the highest number of compromised circuits, and they suggest fixing both the entry and exit ORs as suggested by Wright *et al.* [18].

Abbott *et al.* [1] describe a browser-based attack on Tor where a malicious exit injects a signal generator to the user's traffic. A malicious entry guard is required to perform traffic analysis on its clients' circuits to identify if a circuit carries the injected signal. If such a circuit is identified, then the attacker is able to link the client to its destination. A strong point of this attack is that it does not require both entry and exit to compromise a circuit at the same time, as it only requires that a malicious entry guard detect a specific signal encoded by a malicious web service. The authors argue that using three entry guards helps to protect clients that use honest entry guards. However, the attack becomes more effective for unlucky clients who use malicious entry guards.

Since its initial proposal for Tor, the entry guard design has become more sophisticated, including the many minute details described in Section 2. However, to date, there has been no thorough investigation into the security and performance implications of Tor's entry guard design. This work serves to fill this gap.

7. FUTURE WORK

The major next step, research we are currently engaged in, is to use the results presented here coupled with further COGS-driven analysis to answer the big picture guard flag assignment question posed by Dingledine [8]. A related research problem currently under way is the Tor client model. We noted in Section 3 that it is unclear how to model the Tor client base and the adversary's insertion strategy. We have presented results for the adversary arriving after all clients have picked their guards and no client leaves the Tor network or joins it. Counterintuitive properties—like those in Figure 7 where increasing guard list size actually reduces compromise rates—may not hold for other conditions. Indeed, we have preliminary results, from ongoing research, that suggest that when the adversary exists before the clients arrive, the above counterintuitive properties do not appear. We need better models that accurately reflect user and adversary behaviour in the Tor network in order to properly resolve these questions.

We are also presently considering alternative guard selection algorithms that have desirable properties. As an example of one possible direction, we note that in Section 5 the consensus bandwidth weightings currently utilized to control the guard selection process could be augmented with an age-related weighting that would affect the probabilities of a guard's selection. Also being examined, and closely related, is Tor's "weighted-fractional-uptime" metric—a component in ensuring that the guard flag is given to a relay with little churn—which could be replaced with an alternative calculation that better predicts relay churn behaviour. Another example is a trust-based [13] guard selection scheme such that clients pick guards according to how much they trust them. One final example is to investigate the condition that guards are only added to a client's guard list when fewer than two online guards remain in the list; we have results that indicate that two is a safe value but further analysis remains on how it may interact with various guard selection algorithms.

It is a strength of our COGS framework that it is sufficiently flexible to address all of these, and many related, questions.

8. CONCLUSION

In this work, we investigated the open questions relating to Tor's entry guard design posed by Dingledine [8]. We constructed COGS, a flexible simulation framework, and using it, provided thorough empirical data and analysis to answer most of the questions posed. This leaves the larger question of enhancing the guard selection and allocation algorithms in Tor. This is an area of ongoing work, with promising preliminary results, and we are confident that COGS provides a valuable tool in tackling these important problems.

9. REFERENCES

[1] Abbott, T. G., Lai, K. J., Lieberman, M. R., and Price, E. C. Browser-based Attacks on Tor. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies* (Berlin, Heidelberg, 2007), Springer-Verlag, pp. 184–199.

[2] Bauer, K., and McCoy, D. Tor specification proposal 107: Uptime Sanity Checking. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/107-uptime-sanity-checking.txt`, March 2007.

[3] Bauer, K., and McCoy, D. Tor specification proposal 109: No more than one server per IP address. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/109-no-sharing-ips.txt`, March 2007.

[4] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. Low-Resource Routing Attacks against Tor. In *Proceedings of the Workshop on Privacy in the Electronic Society* (October 2007), pp. 11–20.

[5] Borisov, N., Danezis, G., Mittal, P., and Tabriz, P. Denial of service or denial of security? In *Proceedings of the 14th ACM conference on Computer and communications security* (New York, NY, USA, 2007), CCS '07, ACM, pp. 92–102.

[6] Danezis, G. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)* (May 2004), vol. 3424 of *LNCS*, pp. 35–50.

[7] Danezis, G., and Syverson, P. Bridging and Fingerprinting: Epistemic Attacks on Route Selection. In *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)* (Leuven, Belgium, July 2008), N. Borisov and I. Goldberg, Eds., Springer, pp. 133–150.

[8] Dingledine, R. Research problem: better guard rotation parameters. https://blog.torproject.org/blog/research-problem-better-guard-rotation-parameters, August 2011. Accessed May 2012.

[9] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13* (2004), USENIX Association, pp. 303–320.

[10] Edman, M., and Syverson, P. F. AS-awareness in tor path selection. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009* (2009), E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds., ACM, pp. 380–389.

[11] Murdoch, S. J., and Watson, R. N. M. Metrics for security and performance in low-latency anonymity networks. In *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)* (Leuven, Belgium, July 2008), N. Borisov and I. Goldberg, Eds., Springer, pp. 115–132.

[12] Murdoch, S. J., and Zieliński, P. Sampled traffic analysis by internet-exchange-level adversaries. In *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)* (Ottawa, Canada, June 2007), N. Borisov and P. Golle, Eds., Springer.

[13] Øverlier, L., and Syverson, P. Locating Hidden Servers. In *Symposium on Security and Privacy* (2006), IEEE, pp. 100–114.

[14] Perry, M. Torflow: Tor network analysis. HotPETS, 2009.

[15] The Tor Project. Tor Directory Protocol, Version 3. `https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=dir-spec.txt`. Accessed May 2012.

[16] The Tor Project. Tor Metrics Portal: Data. `https://metrics.torproject.org/data.html#performance`. Accessed January 2012.

[17] Wright, M., Adler, M., Levine, B., and Shields, C. Defending Anonymous Communications against Passive Logging Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy* (2003), pp. 28–41.

[18] Wright, M., Adler, M., Levine, B., and Shields, C. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Transactions on Information and System Security (TISSEC) 7*, 4 (2004), 489–522.