Poster: Guard Sets in Tor Using AS Relationships

Mohsen Imani
UT Arlington
mohsen.imani@mavs.uta.edu

Armon Barton
UT Arlington
armon.barton@mavs.uta.edu

Matthew Wright
Rochester Institute of Technology
matthew.wright@rit.edu

***Abstract*—The mechanism for picking guards in Tor suffers from security problems like *guard fingerprinting* and from performance issues. To address these issues, Hayes and Danezis proposed the use of *guard sets*, in which the Tor system groups all guards into sets, and each client picks one of these sets and uses its guards. Guard sets need regular maintenance, however, due to churn in the Tor network, and this creates opportunities for malicious guards to join many guard sets by merely tuning the bandwidth they make available to Tor. To address this problem, we propose a new method for forming guard sets based on location in the Internet AS topology. We construct a hierarchy that keeps clients and guards together more reliably and prevents malicious guards from easily joining arbitrary guard sets. This approach also has the advantage of confining an attacker with access to limited locations on the Internet to a small number of guard sets. We simulate this guard set design using historical Tor data in the presence of relay-level adversaries, and we find that our approach is good at confining the adversary into few guard sets and thus limiting the impact of attacks.**

I. Introduction

Tor is one of the most popular tools for protecting privacy online. It allows clients to create anonymous connections to their desired destinations via three-hop encrypted channels called *circuits*. A circuit is built over a path of three relays, an *entry*, a *middle*, and an *exit*, selected from among the thousands of volunteer relays distributed across the globe.

Tor fixes the client's entry node to be the same in every circuit for up to nine months. If this entry node, called a *guard*, is honest and does not get compromised, then the client's identity cannot be directly discovered by malicious relays while that guard is still being used.

A key design decision around the use of guards is how to assign guards to users. If a user picks a guard with very low bandwidth, for example, then not only will her performance be poor, she may be the only user regularly using that guard and can thus be profiled [4]. This is known as *guard fingerprinting*. More generally, there are several anonymity and performance considerations for picking guards that have only recently been explored [1], [2].

One solution to the guard fingerprinting problem is to group all guards into *guard sets* [1]. Then, each client picks one of the guard sets and uses the guards in this set for the first hop on all of its circuits. Hayes and Danezis [3] proposed the first guard set algorithm for use in Tor. This algorithm uses guards' bandwidths as the key criterion in forming guard sets, such that all sets have almost the same amount of bandwidth. They also presented techniques for maintaining the guard sets when there is churn in the network. Unfortunately, the algorithms proposed by Hayes and Danezis have vulnerabilities that allow an attacker to compromise many guard sets in the presence of churn over time. In particular, churn among guards leads to guard sets needing to be repaired, which is done by adding guards to the set with approximately the same bandwidth as the guards already in the set. Since the bandwidth that a malicious guard makes available to Tor is under the control of the attacker, he can set his guards' bandwidths so that they get selected for specific guard sets that are about to be repaired. Using this technique, we found (in as-yet unpublished work) that an adversary controlling just 1% of the total guard bandwidth in Tor can infiltrate around 40% of all guard sets within four months.

In this paper, we propose a new design (§II) for guard sets that uses location in the Internet AS topology as the basis for building a hierarchy on top of the sets. In particular, our approach builds and maintains sets using guards that are close to each other in the Internet topology. This limits an attacker's ability to compromise guard sets beyond whatever Internet locations he has access to. While guards' advertised bandwidth can be easily manipulated, most attackers will have a limit on the possible Internet locations of their guards.

II. Design

In our design, a guard's place in the hierarchy and guard set assignment is based on the guard's network location, meaning the Autonomous System (AS) it is in and that AS's corresponding place in the *customer cones*, the set of ASes that can be reached from the root AS by only following provider-to-customer links. For an adversary with high bandwidth capacity and a range of IP addresses, but only a few network locations, this would substantially limit the number of guard sets he can join and the number of users that he can compromise.

The hierarchy consists of three levels: 1) *Root Sets*, 2) *Branch Sets*, and 3) *Guard Sets*. Below, we describe each part of the system in detail.

Root Sets. To form Root Sets, we first build a *Root Set list* – a list of ASes sorted based on customer cone size in ascending order. Initially, the list contains all ASes with one or more guards (*guard ASes*), and each guard AS is considered as a Root Set. Then we choose the Root Set with the smallest customer cone size and follow all customer-to-provider links to discover providers for this Root Set that are also providers to at least one other Root Set in the list. Among the providers that satisfy this constraint, we select the provider with the smallest customer cone size. This provider becomes the new

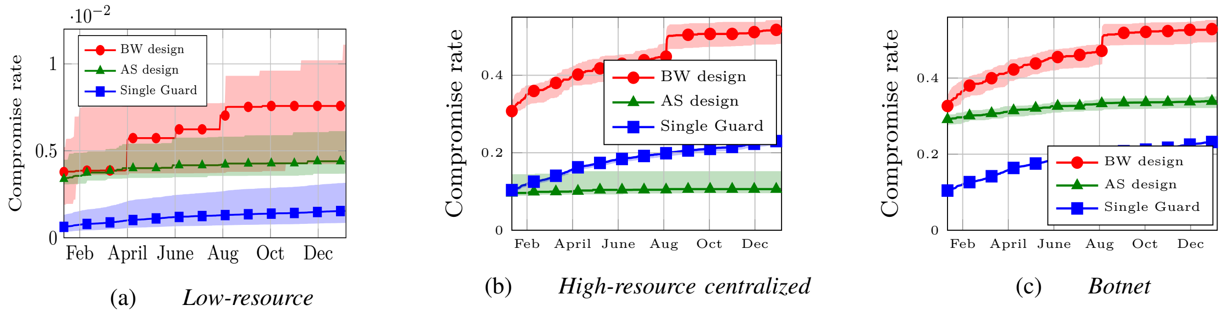

Fig. 1: Compromise rates in different attack scenarios. Lines show the median compromise rate over 50 simulations, while the bands show the first and third quartiles.

Root Set and is added to the Root Set list, while the Root Sets that are in the customer cone of this provider are removed from the list. This process is repeated until all Root Sets in the list contain guard bandwidths that sum to at least a threshold τ_{up} or the number of Root Sets in the list have decreased below a threshold N.

Branch Sets. Root Sets often represent large customer cones and many guards. To better isolate groups of guards from each other and make it harder for a malicious guard to move into targeted guard sets, we break each Root Set into a group of Branch Sets. To this end, we put guard ASes that are close to each other with respect to the AS relationship graph in the same Branch Set as much as possible. For building the Branch Sets, we first identify all customer cones within the Root Set's customer cone in which the guard bandwidth reaches the threshold τ_{up}. Note that some cones will be contained within other, larger cones, and there can be overlaps between cones. Among all the possible customer cones, we should pick cones such that their intersection with respect to guard ASes is empty ($A \cap B = \emptyset$). There may be many possible combinations of customer cones that are independent in this way. Since our goal is to have smaller customer cones to make it harder for an attacker to join a targeted Branch Set, we pick the combination that has the maximum number of independent cones and designate each cone as a Branch Set. At the end, we place all guard ASes that do not meet the requirements for building Branch Sets into one set.

Guard Sets. Once we have the Branch Sets, we can break them up further into Guard Sets. To split a Branch Set, we first randomly shuffle the guard ASes in the Branch Set. We then add one guard at a time from the same AS to the current Guard Set until the Guard Set's bandwidth reaches the threshold τ_{up}. If we use all the guards in a given guard AS without reaching the threshold, we continue adding guards from the next guard AS.

Assigning Clients to the Guard Sets. A newly-joined client first selects a Root Set, then a Branch Set from her selected Root Set, and finally a Guard Set from her selected Branch Set. Each of these selections is random, weighted proportionally by bandwidth. To create a circuit, the client picks one of the guards in her guard set as the entry relay. The selection of guards from a guard set can be weighted in favor of bandwidth or can be done uniformly at random.

Maintenance. We have also designed mechanisms to maintain the hierarchy and guard assignment in the presence of churn, and we elide these for space in this extended abstract.

III. Security Evaluation

In this section, we use simulations based on historical Tor consensus files to evaluate the security of our design. In particular, we model a relay-level adversary who adds guard nodes in the network with the aim of compromising guard as many clients as possible, which means joining as many guard sets as possible.

If a guard set contains one compromised guard, we consider the entire guard set to be compromised, since all clients will select the compromised guard regularly for their circuits. We compare our results to Hayes and Daneziss design [3], which we refer to as "BW design". We call our design "AS design". *Single Guard* refers to the Tor current guard selection mechanism, the single guard proposal. For all of these, we assume that the adversary runs some guard relays such that their total bandwidth adds up to 10% of the total guard bandwidth. Because *AS* design uses both bandwidth and AS relationships, the adversary's network (AS) matters as well. Therefore, we consider three attack strategies:

***Low-resource* adversary:** In this attack strategy, the adversary injects only a single guard relay in the network. We randomly choose an AS for this malicious guard.

***High-resource centralized* adversary:** We assume that the adversary is centralized, meaning that it injects all malicious guards into a single AS. We randomly select one guard AS in which to add the malicious guard relays, and we select their bandwidths randomly based on live Tor guard bandwidths.

***Botnet* adversary:** We assume that the adversary injects his guard relays from different guard ASes instead of one guard AS.

Figure 1 shows the compromise rate in the different guard selection schemes. In all three attacks scenarios, our design protects clients better than *BW* design.

IV. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1423139.

References

[1] R. Dingledine and G. Kadianakis, "One fast guard for life (or 9 months."

[2] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg, "Changing of the guards: A framework for understanding and improving entry guard selection in tor," in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, ser. WPES '12. New York, NY, USA: ACM, 2012, pp. 43–54. [Online]. Available: http://doi.acm.org/10.1145/2381966.2381973

[3] J. Hayes and G. Danezis, "Guard sets for onion routing," in *Proceedings on Privacy Enhancing Technologies*, 2015.

[4] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" *ACM Transactions on Information and System Security*, vol. 13, no. 2, February 2010.

[5] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "Passive-logging attacks against anonymous communications systems," *ACM Transactions on Information and System Security (TISSEC)*, vol. 11, no. 2, 2008.