Henry Tan, Micah Sherr, and Wenchao Zhou

Data-plane Defenses against Routing Attacks on Tor

Abstract: Tor is susceptible to traffic correlation attacks in which an adversary who observes flows entering and leaving the anonymity network can apply statistical techniques to correlate flows and de-anonymize their endpoints. While an adversary may not be naturally positioned to conduct such attacks, a recent study shows that the Internet's control-plane can be manipulated to increase an adversary's view of the network, and consequently, improve its ability to perform traffic correlation. This paper explores, in-depth, the effects of control-plane attacks on the security of the Tor network. Using accurate models of the live Tor network, we quantify Tor's susceptibility to these attacks by measuring the fraction of the Tor network that is vulnerable and the advantage to the adversary of performing the attacks. We further propose defense mechanisms that protect Tor users from manipulations at the control-plane. Perhaps surprisingly, we show that by leveraging existing trust anchors in Tor, defenses deployed only in the data-plane are sufficient to detect most control-plane attacks. Our defenses do not assume the active participation of Internet Service Providers, and require only very small changes to Tor. We show that our defenses result in a more than tenfold decrease in the effectiveness of certain control-plane attacks.

DOI 10.1515/popets-2016-0040
Received 2016-02-29; revised 2016-06-02; accepted 2016-06-02.

1 Introduction

Tor [6] provides anonymous communication to millions of daily users [15, 38] by forwarding messages through a series of volunteer-operated relays. To provide low-latency anonymous communication, Tor does not perform mixing. This allows an adversary who can monitor an anonymous flow both as it enters and leaves the anonymity network to apply statistical techniques to determine that it is observing the same flow, allowing it to de-anonymize the flow's communication endpoints. Such a *traffic correlation* attack [22] can be performed with little cost to the adversary and is dependent mainly on the adversary's ability to observe a flow enter and leave the Tor network [19]. A natural question–and one that has received considerable attention from the Tor community–is *how often is an adversary able to obtain such advantageous positions?*

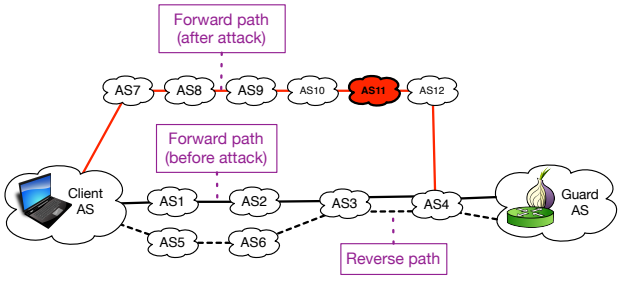

Fig. 1. The forward and reverse routes between a client and its guard relay, shown as black solid and dashed lines, respectively. After a control-plane attack by a malicious AS (AS11), the traffic from the client to its guard is redirected via the route highlighted in red, enabling AS11 to learn the depicted client's network location.

Most of the existing literature that examines traffic correlation considers two potential adversaries: an adversary that operates relays (the *relay adversary*) and an adversary that controls or monitors a region of the Internet (the *network adversary*). The former has a lower bar to entry, since Tor is a volunteer-operated network and anyone can instantiate a relay; examples of the latter include operators of autonomous systems (ASes) or Internet exchange points (IXPs), or nation-states that monitor traffic that traverses their borders. In their analysis of Tor's susceptibility to correlation attacks, Johnson et al. find that moderately-provisioned network adversaries can expect to de-anonymize regular users of Tor (i.e., those who use the service primarily for web browsing) with over 50% probability within three months [19].

Unfortunately, most previously proposed defenses ignore network dynamics, assuming instead (1) a simplified model of the Internet in which routes are static and (2) that adversaries must contend with whatever network position they have. Recent work by Sun et al. [35] show that the realities of dynamic Internet routes and vulnerabilities in the Internet's control-plane protocol (i.e., BGP) lead to yet another avenue by which an adversary may perform traffic correlation attacks. Of particular concern, they show that an AS-level network adversary may interpose itself on traffic between clients and a targeted Tor guard by performing a longest prefix hijacking attack. (A *longest prefix hijacking attack* exploits BGP's longest-prefix matching policy to reroute traffic through the adversary.) Such a scenario is depicted in Figure 1.

This paper explores in-depth the effects of network dynamics on the security of the Tor network. First, we quantify the susceptibility of the Tor network (as it exists today) to

Henry Tan, Micah Sherr, and Wenchao Zhou: Georgetown University

control-plane attacks. To perform our analysis, we construct the most accurate model to date of the live Tor network, inferring AS-level routes using traceroute data from CAIDA [2] and the RIPE Atlas [31]. We study the longest prefix hijacking attacks suggested by Sun et al. [35] and quantify (1) the fraction of the Tor network that is vulnerable and (2) the advantage to the adversary of performing such attacks. We additionally present and evaluate other control-plane attacks in which adversaries advertise (false) shorter paths to a guard. These shortest path attacks may be more attractive to adversaries, since longest-prefix hijacking attacks almost always propagate a previously non-existent prefix throughout the Internet, whereas a shortest path attack is an announcement for existing prefixes but with a shorter hop count and consequently has a more localized effect. As we show below, shortest path attacks are also more difficult to detect, due in large part to their limited range.

Perhaps surprisingly, we show that defenses that rely only on the data-plane are sufficient to detect most control-plane attacks against Tor. We propose and evaluate a defense system for Tor that allows clients to determine which guards are targeted by control-plane attacks, thus enabling them to avoid situations that could lead to a loss of their anonymity.

Our techniques do not aim to solve Internet routing vulnerabilities in general. Rather, we leverage unique features of Tor that make data-plane defenses tractable for the anonymity service: Tor has an existing trust anchor that we can use to provide verifiable messages from relays. In particular, Tor has in place (distributed) authoritative directories that are used to bootstrap the network and allow clients to discover the network locations and public keys of the network's relays. Relays can use these directories to provide signed *descriptors* that describe their views of their local networks. Second, we use the structural properties of the Internet and show that a common topological feature of the Internet, which we call the *subpath property*, can be used to either detect network-layer routing attacks or force the adversary to conduct a much wider-scale attack that affects not only a targeted Tor relay, but also a considerable number of additional ASes. We argue that an adversary who is willing to conduct such an attack will be easier to detect via the collateral damage.

Importantly, our defenses do not assume the active participation of Internet routers or service providers, and can be deployed with very low overhead, requiring only modest changes to Tor's protocols. We impose only additional information in relays' extended descriptors and a few dedicated trusted clients (perhaps operated by the maintainers of the already-trusted directories) that report network-layer routing anomalies.

To evaluate the effectiveness of our defenses, we use our high-fidelity map of the Internet, annotated with Tor relays and popular client and destination locations. We measure the ability of Internet ASes to perform control-plane re-routing attacks against Tor, and quantify the degree of increased Tor ingress traffic that they are able to observe due to their attacks. Next, we compare the adversary's advantage to a scenario in which Tor adopts our defenses. Our results are highly encouraging: for a majority of sampled ASes, we reduce the adversary's view of Tor guard bandwidth due to a longest prefix hijacking attack from approximately 13% of the network's total guard bandwidth to less than 1%, a more than tenfold decrease; for many ASes, we completely mitigate the attack.

In summary, this paper makes the following contributions:

1. We are the first to quantify the adversary's advantage in conducting network-level attacks against Tor. *Key findings*: More than 90% of Tor bandwidth is susceptible to longest-prefix attacks. However, to substantially increase its view of guard traffic, the adversary must be willing to perform multiple prefix hijackings.
2. We introduce a more targeted shortest path based network attack against Tor, derive the topological conditions under which a malicious AS may successfully carry out the attack, and measure the adversary's advantage after conducting a successful attack. *Key findings*: An adversary's ability to perform a useful shortest path attack is highly contingent on its location in the network. However, for many types of ASes, the shortest path attack allows the adversary to see orders of magnitude more Tor guard traffic than it would otherwise, sometimes as much as 15% of the entire network's guard traffic.
3. To support our analyses of network-level attacks against Tor, we develop the most accurate model of the live Tor network to date. We are releasing both the network graphs (in standard formats) as well as the toolchain for graph construction with the publication of this paper.
4. We present a suite of detection techniques which operate in concert to detect network-level attacks against Tor with high accuracy. We formalize the limitations of our defenses, and show that some well-positioned ASes may avoid detection at the cost of causing considerable collateral damage by disrupting large swarths of non-Tor traffic.
5. Finally, we propose a distributed service for detecting longest-prefix attacks against Tor. Our technique uses a small set of distributed monitors, operating only at the application-layer, that carry out our ensemble defenses to detect attacks and enable clients to avoid potentially compromised paths. *Key findings*: Using a modest number of detectors, our distributed defense mechanism reduces an adversary's advantage of network level attack by more than a factor of ten, with few false positives.

2 Threat model

We consider a network-level adversary who operates one or more autonomous systems (ASes). The adversary's goal is to increase its view of anonymous traffic by causing itself to be inserted into the ingress Internet path between clients and targeted guard relays. (We discuss in §8 the symmetry of our defenses against an adversary who attempts to insert itself into the routes taken by egress traffic.) To avoid the trivial case, we assume that the adversary's initial position does not allow it to observe the client's traffic. To achieve its goal, the adversary sends BGP messages to influence the Internet's control-plane in order to alter routes to its advantage.

We mostly concentrate on an adversary who targets specific relays (e.g., those that have the greatest consensus weight and are therefore most likely to be chosen by clients) rather than targeting a specific client or sets of clients. The adversary's aim is to de-anonymize clients en masse—targeting particular clients apriori thus assumes the adversary already knows their network locations. (We briefly consider more targeted attacks in §8.)

We assume that the network-level adversary wants to limit its exposure and stay undetected. We do not limit its ability to announce routes to Tor relays. However, we do assume that the adversary wants to minimize collateral damage caused by its false routing advertisements, and is therefore unwilling to send an unlimited number of BGP messages that adversely affect regions of the Internet that do not host Tor relays. This latter point is relevant, since our defenses are based on observing the effects of the adversary on a large number of ASes (centered around its target), many of which are only used for Internet routing and do not host the targeted Tor relay.

Finally, as we discuss in §3, there are many ways to attack the Tor network (with varying effectiveness), and this paper does not attempt to offer a comprehensive security solution for Tor. Rather, we develop the first defense of which we are aware that counters routing attacks against the anonymity network.

3 Background and Related Work

Tor provides anonymous TCP connections by forwarding traffic from a client (sometimes called the OP for historical reasons) through a *circuit* consisting, usually, of three *relays* (also called ORs). Tor operates as an overlay network, and therefore internal connections within the overlay and ingress (resp. egress) links entering (resp. leaving) the overlay are comprised of potentially multiple network level links spanning multiple ASes (see Figure 1). Each AS that exists along a link in the Tor overlay network is able to observe various properties of the traffic that traverses its network, such as the pair of communicating nodes (e.g., the client and guard, exit and destination, or two adjacent ORs in a Tor circuit) and the distribution and timing of (encrypted) packets [11]. Crucially, an AS who observes ingress Tor traffic can trivially identify the client by inspecting IP headers; similarly, an AS that carries Tor egress traffic learns the network location of the destination. (*Unobservability protocols*, which weaken the adversary's ability to determine that Tor is in use, are discussed below.)

An AS has some control over the traffic that crosses its network, such as imposing rate limits or dropping packets altogether. At somewhat high cost, an AS-level adversary that is able to monitor and affect some portion of Tor's ingress and egress traffic can apply traffic watermarks [16] to correlate flows, and thus de-anonymize anonymous circuits. Even without active traffic alteration, existing work has shown that low-cost statistical analyses applied to ingress and egress traffic are sufficient to de-anonymize the endpoints with high accuracy [27]. As discussed in §1, a relevant question is thus: how often are adversaries situated in positions that enable such traffic analysis? This question has been explored in depth using *static* models of the Tor network [10, 19]. In this work, we measure Tor's susceptibility to traffic analysis against an adversary that leverages the *dynamics* of the Internet's control-plane to actively improve its view of the Tor network.

Defenses against traffic correlation attacks. The problem of defeating traffic correlation attacks has received considerable attention from the privacy-enhancing technologies community, resulting in a number of suggested approaches:

The use of cover traffic and mix cascades [3] hinders traffic correlation, but does so at the cost of enormous bandwidth and latency overhead.

Other work suggests adopting relay selection strategies that avoid untrustworthy ASes [9, 29] or minimize AS crossings [20]. However, as we and others [35] have shown, remote ASes can exploit weaknesses in the Internet's control-plane to change how packets are routed on the Internet, effectively inserting themselves into Tor's ingress and egress paths, regardless of a client's choice of relays.

There has also been considerable research on developing unobservability protocols that hide or disguise a client's use of Tor via traffic morphing [26], steganography [41], domain fronting [12], or format-transforming encryption [8]. However, Houmansadr et al. [17] show that unobservability techniques are ineffective against determined adversaries, since imitation protocols can be distinguished from their true (imitated) protocols using simple probing techniques.

Routing dynamics and attacks. Sun et al. [35] show that Internet dynamics significantly affects the anonymity of Tor users. Naturally occurring churn—that is, fluctuating routing

changes that occur on the Internet's control-plane—and the asymmetry of Internet routes increase the opportunities for a network-level adversary to find itself on paths into and out of the Tor network. Worse, the lack of authentication in BGP enables adversaries to modify network routes to their advantage. In particular, Sun et al. observe that an adversary can conduct a longest-prefix hijacking attack in which it advertises a prefix for a targeted address space that is more specific than the space's currently advertised (correct) prefix (e.g.,1.2.3.0/25 versus 1.2.3.0/24). Since the Internet adopts a longest-prefix matching policy, this effectively causes traffic for the targeted IP address range to be routed through the adversary's AS. An adversary can target the networks of popular Tor guard and exit relays to increase its view of the Tor network, and consequently improve its ability to perform traffic correlation attacks.

This paper measures the longest prefix hijacking attack's effectiveness against the live Tor network. We additionally show that Tor is vulnerable to another similar class of control-plane attacks; these attacks may be especially attractive to an adversary since they may be more difficult to detect than longest prefix attacks. Finally, while Sun et al.'s work focuses on studying the effects of network dynamics and control-plane attacks against Tor, this paper introduces and evaluates low-cost *defenses* against control-plane attacks that can be deployed by Tor clients without requiring access to the Internet's routing infrastructure. The proposed defenses are backed by well-defined heuristics that are drawn and verified by observations of actual BGP data, and offer a principled way towards detection of routing attacks.

BGP defenses and attack detection. The lack of authentication checks in BGP is widely known and is the subject of numerous proposed security enhancements. Control plane solutions such as S-BGP [23], BGPSec [24], soBGP [42], or ps-BGP [40] require significant changes to the routing protocol and the participation in a PKI. The Internet has no such PKI and the ISPs are not incentivized to invest in new router infrastructures to accommodate the required protocol changes. Therefore, these solutions see very limited adoption.

Another class of techniques phrase the problem of detecting BGP misbehavior as a machine learning problem, using BGP updates as inputs to a classifier. For example, Yu et al. [43] propose a cooperate distributed system comprised of ASes to detect anomalous BGP announcements. Similarly, Z. Zhang et al. [45] introduce a distributed detection system that detects and purges bogus route announcements, and J. Zhang et al. [44] apply wavelets to filter out falsified BGP messages. These approaches require the active participation of ASes, and are aimed at protecting ASes from interpreting incorrect announcements. In contrast, this paper assumes the current status quo (i.e., the existence of vulnerabilities in the Internet's control-plane) and proposes methods that allow *end users* to avoid the effects of control-plane manipulations.

There is also existing work that examines this problem of detecting routing misbehavior from only the data-plane. *Secure traceroute* provides authenticated traceroutes, but requires that routers participate in a PKI [28]. *Stealth probing* [1] allows participating end-hosts to detect routing disruptions by embedding probes in IPsec tunnels. However, stealth probes can only detect packet loss and tampering, not malicious rerouting. Such guarantees are already offered in our setting through Tor's inter-relay TLS connections.

Most similar to our work are techniques that employ distributed, traceroute-based IP hijacking detection [35, 46]. Raptor [35] sketches a monitoring framework that detects data-plane anomalies based on traceroutes collected from geographically diverse PlanetLab machines. However, it does not describe how anomalies are detected or how their detection can be used to indicate attacks. As with our approach, Zheng et al. [46] use the subpath property (see §6.1) to detect control-plane attacks and use traceroute to infer AS-level paths between nodes. Both techniques assume a fairly naïve adversary that does not interfere with traceroutes. Such attacks are trivial to mount, since traceroutes are easily identifiable (e.g., by protocol or port, or low TTL) [28] and, once reached, an adversary can forge any downstream path by injecting false ICMP TTL-exceeded messages. In this paper, we also use traceroutes, but consider a knowledgeable adversary that may tamper with measurement probes and inject false responses.

4 Longest Prefix Adversary

Internet routing obeys the *longest prefix match*: if a router has multiple entries covering the same prefix, then it will choose the entry that has the longest (i.e., most specific) matching prefix. This policy leads to a conceptually simple attack: if (i) the AS-level adversary announces a prefix for a target IP space, (ii) the space is not covered by any other more specific announcement, and (iii) this announcement propagates, then Internet routers will choose the path with the longest-prefix (i.e., the one containing the adversary) and the adversary can intercept traffic destined for its targeted IP space.

In the context of Tor, the adversary can target a guard relay by examining the route announcements that cover that relay's IP address and announce a false route with a longer prefix. This causes all traffic destined to that relay to be routed through the adversary, allowing the enumeration of the guard's connected clients. Importantly, this attack can be made mostly transparent to the user, since the adversary can forward packets intact towards the intended guard and prevent the victim

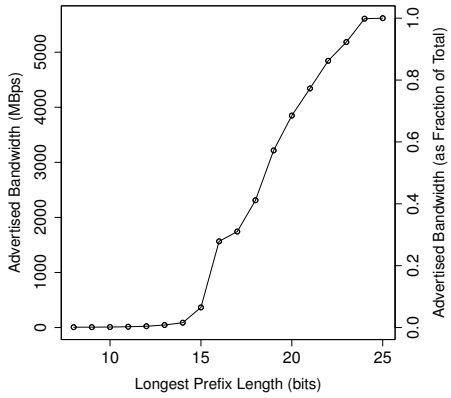

Fig. 2. Cumulative distribution of Tor guard advertised bandwidth (first y-axis) by the longest prefix in the RIB of the routeviews2 RouteViews router (x-axis). The right side vertical axis shows the cumulative advertised guard bandwidth as a fraction of Tor's total advertised guard bandwidth.

clients' connections from being disrupted. (BGP ensures that routing loops will not occur here, since a BGP router will filter out route advertisement whose AS-path already contains its own AS number [30].)

We note that the attack is primarily useful as a means of "repositioning" the adversary to increase its ability to perform traffic correlation. Tor uses TLS between the client (OP) and the guard, and between all Tor relays, using the public key fingerprints in the Tor consensus document to authenticate relays. (A consensus document is a list of Tor relays along with their network addresses and fingerprints of their public keys, signed by Tor's directory authorities.) This prevents the adversary from using control-plane manipulations to undetectably modify Tor streams (e.g., to conduct a MitM attack).

4.1 Overall Vulnerability

We assess Tor's vulnerability to longest prefix attacks using actual Internet routing information from the University of Oregon's RouteViews Project [33]. The RouteViews Project operates multiple Internet routers at various geographic locations and publishes the received route advertisements and *router information bases* (RIBs) of these routers. A RIB contains all the routing information required for the router to route packets at that point in time.

To measure Tor's vulnerability to longest prefix attacks, we extract all of the IP prefixes from the RouteViews router routeviews2. We then take the Tor consensus from the same time period (January 1st, 2014) and find the longest prefix which covers the IP address of each guard. Since longest prefix is the first considered rule when a router is selecting routes [30], we do not need to consider other aspects such as path length.

Figure 2 shows the cumulative distribution of advertised bandwidth (y-axis) of Tor guard relays by the longest prefix known to routeviews2 (x-axis). We measure the impact of potential attacks in terms of relays' bandwidths rather than number of affected relays since Tor biases relay selection proportionally by the relays' advertised bandwidths [6]. Hence, the fraction of guard bandwidth that is observable by the adversary serves as a reasonable estimate of the fraction of Tor clients whose OP→guard traffic is interceptable by the adversary.

The figure shows that over 92% of Tor's advertised bandwidth belongs to guard routers whose IPs are covered by prefixes shorter than /24. Recall that an adversary only has to advertise a route for a prefix that is a longer prefix than all existing advertisements in order to hijack it. As long as the advertisement is not filtered by receiving routers—i.e., it propagates to all other Internet routers—the hijacking will succeed at most points of the Internet.

Most routers allow /24 prefixes as exhibited in Figure 2 (even some /25s are propagated). To prevent highly targeted prefix hijacking, say of a specific /32 IP address, most ASes apply filters that drop advertisements that have prefixes longer than 24 or 25 bits. In summary, this means that over 92% of Tor bandwidth is susceptible to the longest prefix attack.

We note that a prefix hijacking attack with the intention of blackholing traffic should succeed as long as the announcement gets propagated. However, our attacker's goal is to first obtain the traffic then direct it towards its intended destination. This may not always be possible, depending on the choices of paths available to the adversary. For example, if the attacker hijacks the ASes on the true path that the packets take from the attacker to the destination, it may no longer have a way to route packets to the guard. We account for this occurrence in our results below.

4.2 Attack Effectiveness

Although more than 90% of the network's guard bandwidth is vulnerable, actually capturing such a large fraction of Tor's guard traffic requires launching longest prefix attacks against nearly all of Tor's guards (which as of this writing number more than 1600). Clearly, such an attack is likely to have widespread consequences beyond Tor and is very likely to be quickly noticed.

We next characterize the effectiveness of a more realistic attack in which an adversary targets a limited number of selected guards. We emphasize that each guard likely requires its own longest-prefix attack. (Exceptions exist for guards that share long IP prefixes.) Since each attack imposes collateral damage by affecting not just the targeted relay but also all other IP addresses that fall within the targeted IP space, an ad-

versary that wishes to remain undetected will reasonably seek to limit its attacks. As we show in what follows, even a limited attack is sufficient to interpose on a significant portion of Tor's ingress bandwidth.

Modeling the Internet. The effects of BGP manipulation are heavily dependent on the topology of the Internet and the relationships between its ASes. To evaluate an adversary's ability to manipulate the Internet's control-plane to its advantage (i.e., to de-anonymize Tor users) and our ability to defend Tor users from such attacks (see §6), we require accurate models of the Internet's structure and routing behavior.

We construct a topological model of the Internet using traceroute data from CAIDA's Macroscopic Topology Project [2]. CAIDA maintains a set of monitors, distributed across the Internet. These monitors periodically perform traceroutes to the Internet's /24 networks. Each traceroute in the dataset thus provide the path of routers between a monitor and a random destination, from the time at which the traceroute occurred. We create a map of the Internet by stitching together the AS-level paths from the traceroutes from the first week of January 2014. Our map is at the granularity of ASes: each vertex represents an AS and an edge denotes the existence of an AS path as observed from CAIDA's traceroute data. We use MaxMind's GeoIP database [25] to resolve the IP addresses in the traceroutes to their networks' AS numbers[1].

We attach to this graph additional nodes for each relay from the Tor *consensus* of the same time period. These nodes are connected to the AS node to which they belong. We also attach client nodes based on Juen's collection of Tor client statistics [21]. In total, our graph has 15923 vertices (ASes) which cover 4595 relays (1711 of which are guards) and 312 of Juen's noted popular client ASes.

To infer routes, we make use of a shortest AS path heuristic in which we assume that the chosen route for IP packets between any two points on the graph is the path with the shortest number of AS hops. Prior work has shown that the Internet generally routes packets according to the shortest AS path [13]; other work [39] demonstrates that applying a shortest path heuristic over CAIDA's traceroute dataset produces routes that obey the Internet's "valley-free" property [14].

We used the same technique to measure the valley-freeness of our paths: if a path goes from provider to customer, it is not valley free if a future hop goes from customer to provider. Over all paths from clients to guards from the graph where we were able to obtain AS relationships for all ASes along the path using CAIDA's AS relationship database [36], we found that approximately 95% of paths are valley free.

We prune many of the edge ASes in our graph, since they are unable to carry Tor traffic under normal circumstances—for example, leaf ASes (only connected to one other AS) that do not have attached clients, or are disconnected from the main body of the graph. Most of the ASes that we have intentionally excluded would also be unable to perform the prefix hijacking attacks that we describe in this paper. The leaf ASes could hijack routes but would cause a "black hole" as they would then be unable to forward traffic to the target AS.

Since our network graph is useful to study general network problems in Tor (and could be easily used with Tor emulators such as Shadow [18]), we will release the network graphs (in standard formats) as well as the toolchain for graph construction with the publication of the paper.

Accuracy of inferred paths. In addition to valley-freeness of inferred paths, we verify the accuracy of path lengths generated via shortest paths (which returns reasonable candidate paths if not exact path lengths). We obtained traceroutes using RIPE atlas probes [31] between the most popular client ASes and the highest bandwidth guard ASes (and vice versa). Comparing these traceroutes to the routes inferred in our map (using shortest path routing) shows that 75% of inferred paths are at least as long as the actual path. We consider inferred paths which are longer as a training data issue since they occur due to unknown AS links (that appear in the actual traceroutes).

The 25% of inferred paths that are shorter than the actual paths imply that either the graph has improper links, or shortest paths is not sufficient (or both). Further examination of the data showed that more than 95% of paths were no more than one AS longer than that would be predicted by shortest path routing. Additional implications of our shortest path assumption are described in §5.2.

Results. Using our models of the live Tor network, we quantify the adversary's ability to increase its view of guard traffic by conducting a limited number of longest prefix attacks. To select the adversary's targets, we first group guard relays by their ASes and choose the ASes that serve the greatest amount of guard traffic. Conceptually speaking, the adversary targets the IP address ranges that would cause the largest amount of Tor traffic to be rerouted through it. For these results, we assume that all prefix hijacking announcements are propagated to all Internet routers.

The effects of the prefix hijacking attack on the Tor network are shown in Figure 3. The figure shows the client-to-guard bandwidth that the adversary is able to observe (y-axis) as a function of the number of prefix hijacking attacks, against

1 We do not use MaxMind's *geolocation* information. To assess the correctness of MaxMind's *IP-to-ASN mappings*, we compare it with the map provided by CAIDA [32]. The CAIDA map is itself based on RouteViews data (i.e., routers' RIBs). We found that the two mappings generally agree, with only about 1% disagreement over the IPs that constituted our Internet model. We chose the MaxMind database since it is packaged with Tor.

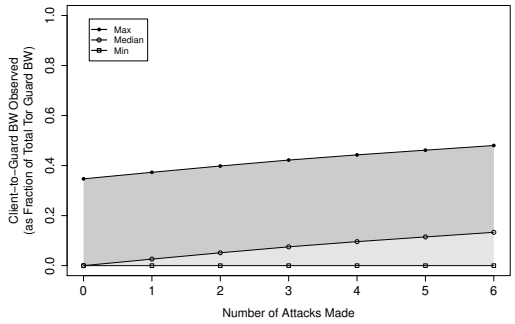

Fig. 3. Maximum/mininum/median Tor ingress bandwidths that different ASes performing zero to six attacks can observe. Maximum ingress bandwidth seen is the largest ingress bandwidth seen by any attacking AS over all possible AS attackers, similarly for minimum. These values include the Tor ingress bandwidth they could already observe before they perform the attacks (which corresponds to when the number of attacks is zero).

different /24 prefixes, the adversary is willing to perform (x-axis). In our analysis, we consider all possible ASes as potential adversaries. We remark that based on their positions in the Internet, some adversaries will be already predisposed to observing some fraction of Tor's client-to-guard traffic.

In the absence of an attack, most adversaries (i.e., ASes) are not able to observe any Tor traffic. This is perhaps unsurprising, given that most ASes exist along the Internet's edges (e.g., customer ASes), with a much smaller portion constituting the Internet's core. With a single prefix hijacking attack, however, the majority of ASes on our graph could observe approximately 2.6% of Tor's client-to-guard traffic. As expected, the effect is increased when the adversary is more willing to attack additional IP prefixes. When the adversary is willing to carry out six longest prefix attacks, she can improve her view to approximately 13.2% at the median, and 48% at the maximum, of Tor's ingress traffic.

5 Shortest Path Adversary

The *shortest path adversary* advertises a fake path from itself to the target prefix that has fewer AS hops than its actual shortest path to the target. An example is shown in Figure 4. Neighbors of the attacker (up to some distance from the attacker) will find this path more attractive and potentially divert traffic destined for the target towards the attacker.

This attack takes advantage of the shortest AS path routing rule used by many BGP routers. When a BGP router following the *shortest AS path* rule has multiple advertisements to the same prefix, it tends to pick an advertisement with the fewest number of ASes in it (see §4.1). Without knowledge of the underlying physical links, this attack is difficult to detect since legitimate AS advertisements and actual paths on the Internet do experience churn and change over time.

5.1 Attack Details

Given an adversary AS A and the target's AS T (i.e., one that contains one or more guards), we allow the adversary to make a prefix hijacking route advertisement for some prefix in T which is as short as possible but still not so short that it causes a *black hole* (explained below). The shortest AS path heuristic that we use to infer paths is critical since the attacker is taking advantage of exactly this routing rule to hijack the prefix.

Let $p_{x,y}$ be the path from AS x to AS y that contains y but not x. $p_{A,T}$ is then the actual path from the adversarial AS A to the target AS T (where the Tor guard resides) with length $|p_{A,T}|$. In order to attract more traffic destined for the target towards itself, the adversary advertises some path p' with length $|p'| < |p_{A,T}|$. We will now show that $|p'| \geq \frac{|p_{A,T}|}{3}$ and that p' must contain some of the ASes from $p_{A,T}$, specifically the ASes which show up in the first third of the path.

First consider an adversary which claims a path length of one hop, i.e., that it is a neighbor of the target. This attack will have the most effect on the Internet, via the shortest AS path routing rule, relative to path lengths of other hop counts. However, Figure 4 (left) shows that the adversary will effectively cause a black hole to occur by way of a cycle in the graph since traffic going through its original route $p_{A,T}$ will be directed towards it. This is a problem for our adversary since its goal is to perform a correlation attack and thus it needs to be able to re-direct the traffic it receives to its actual destination.

Assuming the adversary controls only its AS A, it can prevent the black hole from occurring by preserving its true path $p_{A,T}$—that is, by preventing the ASes on $p_{A,T}$ from preferring its false path announcement.

The first way the adversary can cause nodes in $p_{A,T}$ to ignore its announcement is to make p' longer and less attractive. We know that for a given $|p'|$, the nodes in $p_{A,T}$ that are sufficiently close to the target will be unaffected by p'. Firstly, note that each successive node u in $p_{A,T}$ has a path $p_{u,T}$ that is one hop shorter than that of the previous node. Secondly, we know that the path from a node $u \in p_{A,T}$ to the target through the adversary, i.e., if it were to receive the fake announcement, is perceived by u as having length greater or equal to $|p_{u,A}|+|p'|$. This is illustrated in Figure 4 (right).

Therefore, for a given $|p'|$, a node $u \in p_{A,T}$ will choose its original (correct) path if

$$|p_{u,T}| < |p_{u,A}| + |p'| \tag{1}$$

If we were to consider only this restriction, the adversary would be forced to advertise a path such that $|p'| \geq |p_{A,T}| - 1$

Fig. 4. Example scenarios of shortest path attack, where the attacker AS (highlighted in red) sends false announcement to its neighbors to attract traffic. The actual path from the attacker AS to the guard AS is denoted by solid lines, and the fake paths announced by the attacker are denoted by dashed lines. The ASes that are influenced by the false announcement are highlighted in bold.
In the left scenario, upon receiving the false announcement, AS_{a1} will change its path to the guard AS and route the traffic through the attacker AS. This will then nullify the actual path from the attacker AS to the guard AS, and cause route oscillation or even loss of reachability. Here, the adversary is unable to perform traffic correlation. In the right scenario, however, a knowledgeable attacker can enforce $AS_{f1} = AS_{a1}$, in which case, none of the ASes on the actual path from the attacker AS to the guard AS will be affected, either because its original path to the guard AS is shorter (AS_{a2} and AS_{a3}) or the false announcement is filtered out on loop detection (AS_{a1}).

since the first hop of $p_{A,T}$ would choose the wrong path for a shorter p'. We thus also make use of the routing rule where a BGP router will ignore advertisements for a prefix if the path in that advertisement already contains its own AS [30].

As such, A can include in p' the closest ASes in $p_{A,T}$. This implies that for all u not in p', $|p_{u,A}| \geq |p'|$. Using this and the substitution $|p_{u,T}| = |p_{A,T}| - |p_{u,A}|$, we can solve Eqn. 1 to yield $|p'| \geq \frac{|p_{A,T}|}{3}$.

This shows that the advertised false path must have a minimum length, and must contain certain elements based on the actual topology of the Internet.

One assumption that we make is that no other attack affecting ASes in p is occurring. If this is not the case, a cycle might occur since this attack relies on the very specific path $p_{A,T}$ from being unperturbed. More generally, if $|p_{u,A}| < |p_{A,u}|$, for reasons including other attacks or path asymmetry, the attack could cause a cycle.

5.2 Susceptibility to Shortest Path Attacks

To examine the effectiveness of the shortest path adversary, we use our model of the Internet that is described in §4.2. We emphasize that since we have no visibility into ASes' local routing preferences and our map relies on a shortest path heuristic (which we show in §4.2 to be mostly accurate in practice), the results in this section should be considered conservative. When ASes have local preferences that choose non-shortest routes, the attacks will *not* be effective. That is, our analysis below assumes the normal (but not universal) policy of routing according to shortest AS paths.

For each ⟨attacking AS A, guard AS T⟩ pair in the Internet map, where we allow any AS to be an attacking AS, we alter the map such that routes are effectively altered as though A had advertised a false path to T of length $|p_{A,T}|/3$. By the analysis in §5.1, we know that all ASes along $p_{A,T}$ will ignore this advertisement.

We then calculate all routes from all clients to T to find out which ones now have their traffic routed through A, which did not previously go through A. From this and the client traffic distributions from Juen [21], we are able to calculate the amount of Tor bandwidth that the attacker was able to observe before and after the attack.

Figure 5 plots the cumulative distribution of the increase in ingress Tor bandwidth (as a fraction of Tor's total guard bandwidth) observed by the adversary due to the shortest path attack. Points on the curve are from all potential adversaries (i.e., all ASes on our map) and assume intelligent selections of guards such that the effects of the attack are maximized.

The results confirm that an adversary's ability to perform a useful shortest path attack is contingent on its location in the network. As one might expect, the majority of ASes are not able to significantly increase their view of the Tor network, even if they launch attacks against six Tor guards. However, for a large number of ASes, manipulating the control-plane via shortest path attacks yields a considerable view of Tor's ingress traffic. For example, for approximately 15% of ASes, launching two shortest path attacks enables them to view roughly an additional 5% of all Tor ingress traffic. Figure 6 provides a slightly different perspective, showing in log scale the percentage of Tor traffic viewable by each AS, before and after it launches an attack on six targeted guards. Missing red data points indicate that the AS could not observe any pre-attack bandwidth (and are thus missing due to the log plot).

We additionally study the relationship between the type of AS and its ability to perform useful shortest path attacks. Using the AS categorization data from CAIDA [2], we show in Figure 7 that different types of ASes have different advantages in terms of hijacking. ASes that provide home Internet access (*"access"*) or host content or content distribution networks

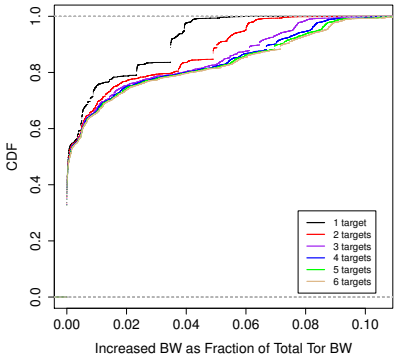

Fig. 5. The CDF of ASes by amount of potential increased Tor guard bandwidth observed for different numbers of target ASes. Each AS adversary targets one or more guards, chosen such that the attacks result in the largest increase in observed bandwidth.

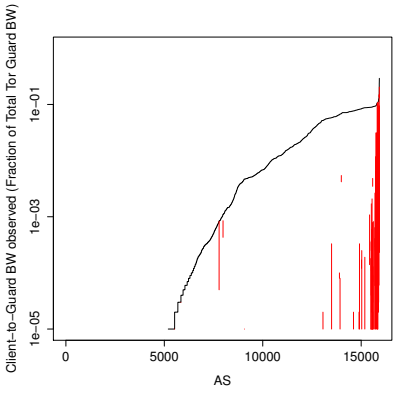

Fig. 6. The potential fraction of total Tor bandwidth each AS can observe before (red) and after (black) performing attacks on the six other ASes which give them the most bandwidth. The ASes on the graph are sorted by the amount they can observe after performing their attack.

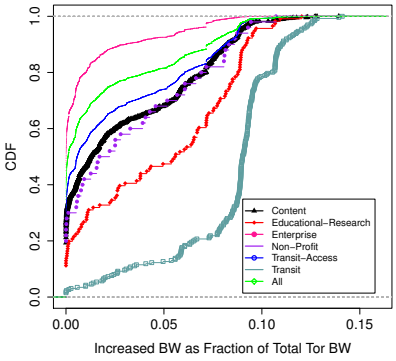

Fig. 7. The CDF of ASes by amount of potential increased Tor guard bandwidth observed for different categories of ASes. Each AS adversary targets six guards, chosen such that the attacks result in the largest increase in observed bandwidth.

(*"content"*) are able to hijack the most bandwidth, which is not surprising given that they are located closer to the "core" of the Internet and are thus well situated to shorten their paths while still leaving their original paths to the targets unaffected.

For ease of exposition, we refer to control-plane attacks generally as *prefix hijacking* in the remainder of the paper. When relevant, we will distinguish between longest prefix and shortest path attacks.

6 Defenses

We introduce multiple detection techniques which operate in concert to identify guards that are the targets of IP prefix hijacking. To achieve our goals using only techniques that operate on the data-plane, we leverage structural properties of the Internet's topology and Tor's existing trust infrastructure.

A high-level overview of our approach is presented in Figure 8. Briefly, guard relays discover their neighboring ASes via traceroutes, and attest to those neighboring ASes by listing them in their Tor descriptors. A set of trusted clients which we call *control-plane authorities* (CPAs[2]) that are operated by maintainers of the Tor Project (or their trusted designates) periodically check whether guards are the targets of a control-plane attack. CPAs perform this check by running traceroutes to the guards and requesting that the guards run traceroutes back to them. (Importantly, these traceroutes may be manipulated by an adversary.) CPAs verify that (i) the AS neighbors of the guard (as derived from the CPAs' traceroutes) are attested to by the guard, (ii) that the AS hop counts of the traceroutes to/from the guard are approximately equal, and (iii) that AS-level paths from the CPAs to the guard exhibit a topolog-

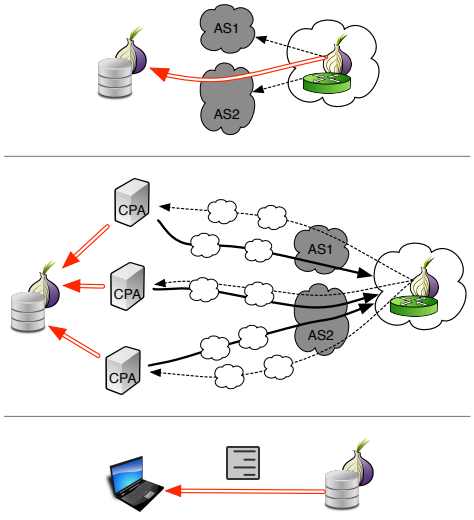

Fig. 8. *Top:* The guard performs traceroutes (dotted arrows) to determine its immediate AS neighbors (AS1 and AS2), and enumerates these neighbors in a descriptor it shares with the Tor directories (red double-arrow). *Middle:* A small number of CPAs and the guard conduct traceroutes. The CPAs then verify (i) the reported neighbor AS of the guard is one that was reported by the guard, (ii) that the subpath property holds and (iii) that the AS hop count in their traceroutes matches the hop count in the guard's traceroutes. Finally, they report their evaluation securely to the directory authority (red double-arrow). *Bottom:* A client fetches a signed Tor consensus document from the Tor directory authority (or one of its mirrors) and considers only the guards that have the SAFEGUARD flag.

2 Admittedly, this is a misnomer since the CPAs have access only to the data-plane. Given their function, we think the CPA acronym is fitting.

ical property called the *subpath property* (explained in §6.1). If a (tunable) fraction of CPAs certify that a given guard relay passes all three checks, then Tor's directory authorities assign that relay a special SAFEGUARD flag and publish the flag in its consensus documents. During relay selection, Tor clients avoid guards that do not have the SAFEGUARD flag.

We first cover each component of our defense, with data to show their reliability and limitations (§6.1, §6.2, and §6.3). We then describe how we combine our defenses and, through both analyses and experimentation, show how each part protects against various capabilities of the adversary (§6.4). Finally, we propose a distributed defense mechanism for Tor that uses the techniques described in §6.1-§6.3 to enable clients to learn which guards to avoid.

Notation. In the remainder of this section, we slightly revise our earlier notation to let $p_{A,B}$ indicate an Internet AS path *as determined by traceroutes* from source A to destination B. Let $|p_{A,B}|$ be the number of ASes in path $p_{A,B}$.

6.1 The Subpath Property

The *subpath property* states that the Internet path from a source x_1 to a destination x_n and the hops on that path share certain characteristics. Formally, given an AS-level path p_{x_1,x_n} of the form $p_{x_1,x_n} = x_2, x_3, \ldots, x_n$, a path p_{x_1,x_k} for $x_k \in p_{x_1,x_n}$ should be $x_2, \ldots, x_k$, which is a prefix of p_{x_1,x_n}. The subpath property was first observed and used as a detection mechanism for prefix hijacking attacks by Zheng et al. [46].

When a prefix hijacking attack is successful against a target destination D, it must hold that the path $p_{C,D}$ from a client C either contained the adversary prior to the attack, or now contains the adversary where it did not before. We do not address the former case since the client was already compromised before the attack. For the latter, the path $p_{C,D}$ has changed. Crucially, however, paths to non-hijacked prefixes will not change. If we assume that the subpath property holds under normal circumstances, then if it is ever violated, we can assume that a prefix hijacking attack is underway.

For a client to check the subpath property of its path to a guard node, it first runs a traceroute to the guard. It then selects some IP address from the path, that is in an AS that is different from the one the guard node is in, and runs a traceroute to that IP address. The client then checks if the AS path of the latter traceroute is the expected prefix of the former traceroute.

Example. Consider the paths shown in Figure 1. Before the attack, the client obtains the AS path ⟨AS1, AS2, AS3, AS4, Guard AS⟩ via a traceroute to the guard. If it then runs another traceroute to an IP selected from the original traceroute that belonged to, say, AS3, then it will obtain the AS path ⟨AS1, AS2, AS3⟩, which is a subpath of the original traceroute. However, after the attack targeting the guard's AS, the client's traceroute to the guard shows the path ⟨AS7, AS8, AS9, AS10, AS11, AS12, AS4, Guard AS⟩. (For now, we assume that the adversary at AS11 does not manipulate the traceroutes that pass through it.) If the client then performs a traceroute to a router in AS4, it obtains the path ⟨AS1, AS2, AS3, AS4⟩, indicating a violation of the subpath property. The key observation is that attacking a prefix *only* affects routes towards that prefix.

Subpath target relative to dest.	Fraction of compatible cases
-1 (AS neighbor of dest.)	0.897
-2 (two AS-hops from dest.)	0.954
-3 (...)	0.974
-4	0.961
-5	0.926

Table 1. The fraction of subpaths (right col.) for a given distance relative to the destination (left col.) that are compatible with the subpath property.

Verifying the diffuseness of the subpath property. To empirically verify that the subpath property is present on the Internet, we ran traceroutes from 132 different VPN based vantage points, all located in different ASes, to the top 300 most likely chosen guards. We use the unfortunately named hidemyass.com VPN service due to its large number of geographically diverse VPN exit points. For each of the traceroutes, we checked whether the subpath property holds at the i-th AS hop of the path starting from the end. We also call the i-th AS hop from the end of a path the $(-i)$-th AS hop of a path. The results of this experiment, found in Table 1, show that if one were to test the subpath property using the (-2)-th hop, for example, that it would hold 95% of the time. Here, we consider two paths to be *compatible* with the subpath property if they could match each other even though they both may contain non-responding routers in different hops, which induce ambiguities. We replace these hops with wildcards when we test for compatibility.

One of the main problems with the subpath defense is its reliability on traceroute which is an insecure technique that relies on packets which, in the case of an attack, pass through the adversary and may be freely manipulated by the adversary. As such, we require other methods to be used in tandem to detect instances in which the adversary manipulates traceroutes.

6.2 Hop Count Restriction

It is fairly well known that paths on the Internet are not generally symmetric (by IP and AS). We confirm this for the Tor network using 100 RIPE Atlas probes in the most popular client

and guard ASes that are covered by the RIPE Atlas probe network [31]. As before, we use Juen's study [21] to determine popular client ASes and Tor's consensus information to obtain the guards with the highest selection probability. We find that 54.5% of paths are asymmetric at the AS level. However, we find that a more relaxed restriction does hold: the number of ASes in a path $|p_{a,b}|$ is close to $|p_{b,a}|$.

We empirically verify this property by using the same traceroutes from the RIPE probes. The results, displayed in Figure 9, show that for more than 90% of the paths between the tested client/guard pairs $\langle C, G\rangle$, that $\text{abs}(|p_{C,G}| - |p_{G,C}|) \leq 1$, where $\text{abs}(\cdot)$ denotes absolute value.

Assuming this property, the client could request a traceroute from the guard and compare it to its own traceroute. If it finds that the hop counts differ too greatly, i.e., $\text{abs}(|p_{C,G}| - |p_{G,C}|) > 1$, it detects that the guard may be under attack.

Consider an attacker A which does not manipulate traceroutes. When A performs the prefix hijacking attack, assuming the shortest AS path heuristic on the Internet, the resulting paths to the guard will be at least as long as the original path. For example, returning to Figure 1, the AS hop count from the guard to the client (as provided by the guard) is five, while the client's measured hop count to the guard (post attack) is eight, a difference that signifies a large departure from the norm.

Pairing this technique with the subpath property increases the accuracy of detection, or reduces the false positive rate, depending on how sensitive the client is to prefix hijacking.

Now consider a more intelligent adversary that drops all traceroutes that pass through it, or even just traceroutes towards guard ASes. Since traceroute packets can be spotted without deep packet inspection (varying TTLs from the same source IP address), an AS adversary can easily filter out and drop traceroute packets. This will effectively reduce the number of AS hops perceived by the client when it performs a traceroute to the guard, evading the hop count check. Additionally, the adversary could potentially forge ICMP TTL exceeded messages from AS hops after itself to increase $|p^*_{C,G}|$ to the right value. As such, we consider a less strict inequality, $|p_{C,G}| \leq |p_{G,C}| + 1$, as the one which might be violated and use this later in our analysis. This effectively assumes the adversary is intelligent enough to pad a client's traceroutes to the right number of AS hops. Hence, we require additional techniques to detect such manipulation.

6.3 Relay Neighbor-AS Discovery

We now describe a third component of our defense which provides improved effectiveness when used in tandem with the subpath property (§6.1) and the hop count restriction (§6.2).

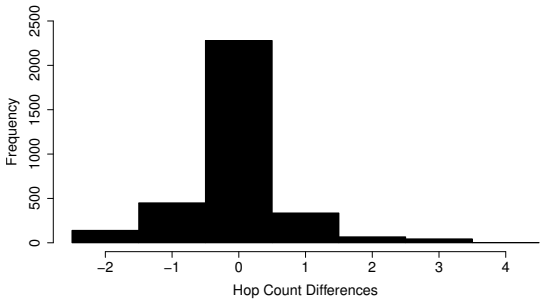

Fig. 9. Histogram of the differences between the AS hop counts of client→guard and guard→client traceroutes.

The guard node identifies its neighboring ASes, out to some number (k) of AS hops, and publishes this information as *Extrainfo descriptors*—a document containing attributes about a relay that can be retrieved by Tor clients and verified through Tor's existing trust infrastructure. To find its AS neighborhood, a guard can periodically run traceroutes to random /24s and enumerate the ASes it observes. Since guards' locations on the Internet are unlikely to change often, this can be performed infrequently. However, as we show in §7, only the 1-hop AS neighbors are needed in practice to achieve high detection rates—this may be leveraged to significantly reduce the overheads of our defense. To reduce the number of traceroutes required to discover AS neighbors, the guard can instead run a traceroute to an IP address from each AS (as opposed to each of the 2^{24} /24 networks) on the Internet. CIDR shows that there are currently less than 50,000 unique ASes [5], and hence a complete scan requires only a few megabytes of bandwidth (see §7 for a fuller discussion of bandwidth overheads).

When considering a guard, a client retrieves the guard's AS neighborhood information from the Extrainfo descriptors maintained by the directories. The client then runs a traceroute to the guard, verifying that the reported ASes are found at the end of its traceroute. If they are not, the guard is not used.

Importantly, this alone is effective against only naïve adversaries since more clever adversaries can forge entries for downstream ASes when traceroutes are routed through them. That is, an adversary can falsely indicate that the path through it terminates with ASes that are attested to by the guard.

Information leakage. When this defense is deployed, the client gains some reliable information on what AS hops it should expect to be able to use as test points to compare the subpath property. If the adversary manipulates traceroutes passing through it, it is now restricted to revealing the correct ASes at the end of the client's traceroute (else it is trivially discovered by the defense). The adversary must either (i) launch an additional control-plane attack against the AS-neighborhood of the guard (i.e., so that it can also appear in the subpath traceroutes to those neighbors) or (ii) guess ASes for

each client such that the traceroutes for the client already go through the adversary, and those guessed ASes intersect with the AS neighborhood of the guard. We posit that the second scenario is highly unlikely for most ASes.

We note that this defensive measure may reveal information to the adversary, even if the attack is detected: suppose that a client C considers a guard G that is the target of a prefix attack from adversary A. The defense requires C to perform a traceroute to G, which will flow through A's AS, revealing C's network location—even if C detects the hijacking and aborts its use of G. Although this prevents A from performing traffic correlation (since C will not form circuits through G), it allows an adversary to perform an enumeration attack in which it seeks to learn the network locations of Tor users. We return to this issue in §7, where we present a solution that enables clients to learn whether a given guard is targeted by a longest-prefix attack without having to contact that guard.

6.4 Analysis of an Ensemble Defense

The detection techniques described in §6.1-§6.3 can be combined into an ensemble defense strategy. We next analyze the security of this ensemble defense against both longest prefix and shortest path adversaries.

Longest prefix adversary. The simplest adversary A that we consider is one that (a) controls a single AS, (b) performs a prefix hijacking attack on a single prefix that contains one or more guards, and (c) does nothing to protect itself from detection (i.e., it does not intercept or manipulate traceroutes).

We consider a client C that selects a guard node G in the hijacked prefix, where A was not on the path $p_{C,G}$ prior to the attack but is on the compromised path $p'_{C,G}$ after the attack has started. Without running any defenses, C's traffic is compromised by the attack. As mentioned in §6.1, C can check the subpath property on $p'_{C,G}$ to detect this adversary. Recall that this naïve adversary is detected since it is not on the paths to the Internet routers that appear in $p'_{C,G}$.

We now allow A to manipulate all packets that it receives. For example, A may drop all traceroute packets going through it, or even forge the ICMP replies so that they appear to come from ASes such that the subpath property holds when checked by the client. Suffice it to say, once a cleartext packet goes through the adversary, we should make no assumptions on the correctness or authenticity of the packet past that point. We define $p^*_{C,G}$ as the path from C to G as observed by C's traceroute to G, which may have forged hops in them.

The hop count check described in §6.2 can be used in this adversarial model. The client checks whether the $p^*_{C,G}$ path is longer than the $p_{G,C}+1$ path to detect A. A is unlikely to be on the path $p_{G,C}$ so it is only able to manipulate $p^*_{C,G}$. One important point to note is that the attacker must respond with a path $p^*_{C,G}$ that satisfies $|p^*_{C,G}| \geq |p_{C,A}|$ since it is unable to manipulate packets that have not passed through it, in particular, the ones in the path $p_{C,A}$ prior to A. This is based on the assumption that the adversary is restricted by the physical connectivity of the Internet as it cannot move itself in the network or form arbitrary physical links. Even with cooperation from other ASes, the hop count restriction still holds up till the first time one of the cooperating (and now adversarial) ASes receive the traceroute packets.

It is worth emphasizing that with this defense, even if the adversary performs prefix hijacking on the intermediate ASes on the path from $p_{A,G}$ to ensure the subpath property holds, the above techniques will still detect the attack. Intuitively, the adversary's position in the network and the AS hop count restriction limit its ability to produce a believable traceroute.

The client can partially get around the problem of inaccurate information from the adversary by utilizing the guard AS neighborhood defense from §6.3. If the diameter, in hop count, of the AS neighborhood published by the guard is k, the adversary must add k hops to $p^*_{C,G}$ to avoid detection. We note that in the case that the adversary is in the k hop neighborhood, it will be able to add fewer hops. For small k however, the number of ASes that are in the k neighborhood of a particular guard is small. For the more general case, this improves the AS hop count inequality detection test to $|p^*_{C,G}| \geq |p_{C,A}| + k > |p_{G,C}| + 1$.

Unfortunately, even with this improvement, not all clients are situated in the Internet for the inequality to be satisfied. But while the adversary may be able to fool some clients all the time, it is unlikely able to fool all clients—a fact we leverage in our distributed detection mechanism, described in §7.

Shortest path adversary. Up till now, our analysis has considered only a longest prefix attacker. The hop count defense does not detect the shortest path attacker for the same reason that the shortest path attacker is unable to propagate its false announcement and have it accepted by as many ASes as the longest prefix attacker. The only clients which are susceptible to the shortest path attack, and therefore have a chance to detect it, are the ones for which $|p_{C,G}| > |p_{C,A}| + |p_{A,G}|/3 > |p_{C,A}|$. These are the clients for which the AS neighborhood published by the guard must be quite large, and we suspect large enough that it becomes both infeasible to complete and not as useful since it will contain far too many ASes.

If we restrict the shortest path attacker to attacking only guard ASes, the subpath property will still be able to detect him. As such, both defenses should be used together.

An important limitation of our work is that a shortest path adversary who is willing to launch additional prefix hijacking attacks against the guard's neighboring ASes may avoid detec-

tion. However, we note that such attacks introduce collateral damage by attacking not just the guard but also the routers that belong to ASes that neighbor the guard, and are therefore likely to be noticed using other means. Additionally, we re-emphasize that, unlike longest prefix hijacking attacks that affect large swarths of the Internet, shortest path attacks affect only a limited number of (nearby) clients.

7 A Distributed Service for Detecting Longest-Prefix Attacks

We introduce a distributed protection mechanism for Tor that leverages an adversary's inability to simultaneously avoid detection from probes situated at different vantage points on the Internet. Our technique is most effective against longest prefix hijacking attacks and relies on a small number of such probes that we call *Control-Plane Authorities* (CPAs). We envision CPAs being operated by the Tor Project or by their designates. (Similar schemes that rely on distributed trusted parties already exist in Tor, for example, to monitor the bandwidth of relays [37] and to discover malicious exit relays [34].) In essence, the CPAs perform the ensemble defense described above and report their findings to Tor's directory authorities. If the fraction of CPAs that detect control-plane manipulations is under a threshold value, then the directories assign it a SAFEGUARD flag, indicating that it is deemed safe for use.

We use the model of the Internet, described in §4.2, to evaluate our distributed defense against a longest prefix hijacking attack. We place a CPA in each of the top 20 client ASes (as reported by Juen [21]) and allow the adversary to attack any six /24 prefixes.

Experimentally, we find that when a guard attests to its single-hop or two-hop AS neighborhoods, our defense achieves strong results, as shown in Figure 10 (left and right). We select a detection threshold of eight—that is, if eight or more CPAs report that a given guard is under attack, the attack is considered detected. (We discuss the selection of this threshold later in this section.) We obtain the bandwidths that one adversary would be able to observe if it attacked the six guards that would give it the most Tor bandwidth. For these results, we sample uniformly at random 800 of the ASes in the network map for the adversary. We conservatively assume that the adversary knows when it will be detected if it were to attack a guard, and will only choose guards where it will not be detected. That is, we assume not only that the adversary knows the locations of the CPAs, it also has a perfect representation of both the Internet's topology and routing policies.

Our results show that when our distributed detection technique is not applied, a prefix hijacking attack against six targets yields about 13% of all Tor guard bandwidth for almost any AS. We consider two versions of the defense, one where the guards announces a one-hop neighborhood (Figure 10, left), and one where the guard announces a two-hop neighborhood (Figure 10, right). While the two-hop neighborhood takes more effort to probe, the detection mechanism reduces attack effectiveness to almost 0.5% for most of the ASes and the rest are reduced to at most 4% of all Tor guard bandwidth.

Bandwidth overhead. The bandwidth cost for each CPA is fairly modest at 16.32 MB for each detection cycle. Each traceroute packet is 80 bytes, and the default max TTL of a traceroute is 30. As such, the cost of a single traceroute is at most $2 \cdot 80 \cdot 30 = 4800$ bytes and is typically smaller. The factor of two encompasses both the outgoing and incoming bandwidth. For the subpath property technique, the CPA runs traceroutes to both the guard and to its neighbor. Therefore the total bandwidth cost for a CPA to test all the guards, which is just under 1700 at the time of this writing, is approximately $4800 \cdot 2 \cdot 1700$ bytes = 16.32 MB per detection cycle. For a detection cycle of once per hour, for example, the bandwidth overhead is just 4.53 KB/s. CPAs can be set to probe different subsets of the guards, further reducing their workloads.

The cost of this process is even lower for each guard. Guards have to perform one traceroute per CPA, and receive one traceroute per CPA. Given a 20 CPA setup, this costs $2 \cdot 4800 \cdot 2 \cdot 20 = 0.384$ MB per cycle. The AS neighbor discovery process where a guard runs traceroutes to a single IP address in each AS, assuming a cutoff TTL of 4, takes approximately 32 MB. Since AS neighbors are unlikely to change often, if run once a day, this process requires 0.37 KBps.

The overhead added to the Extrainfo descriptors is small: 10 to 15 bytes per guard relay, without compression (we found that guard ASNs typically do not have a large number of AS neighbors). As of this writing, Tor has 1666 guards, resulting in an overall overhead (across all guards) of just 24.4 KB.

Resilience to churn. Sun et al. [35] showed that BGP churn is an important factor to consider when measuring the threat to Tor at the AS level. We describe here why our data and techniques are minimally affected by churn (if at all).

Consider the separate parts of our defense:

1. The subpath property (§6.1) requires multiple traceroutes by a CPA towards one guard. By design, the CPA will perform these traceroutes in quick succession, making the probability of BGP churn affecting the results very low.
2. For the hop count restriction (§6.2), the CPA runs a traceroute to the guard to obtain the hop count. It then requests that the guard perform a traceroute back to itself and compares the hop counts. Assuming the guard replies promptly, BGP churn should not affect the comparison.

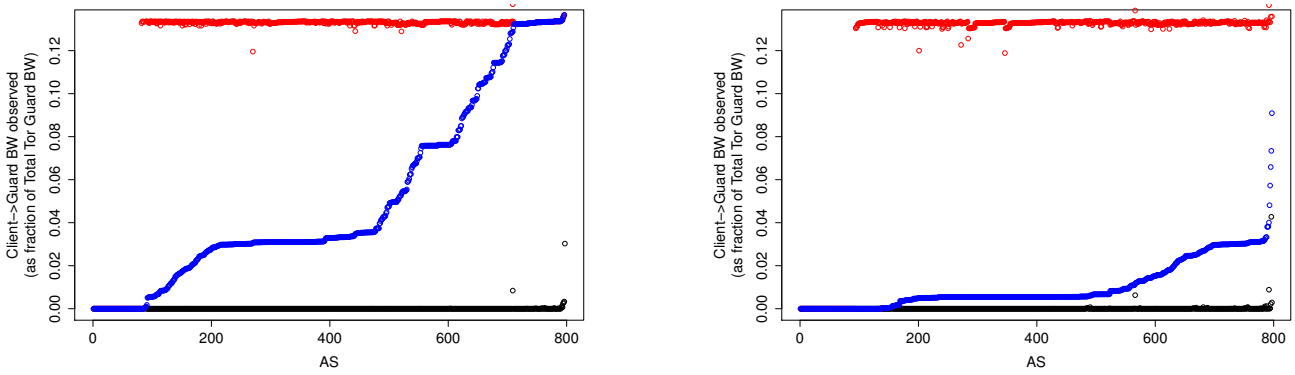

Fig. 10. Fraction of Tor bandwidth observed by different adversarial ASes. The points indicate the amount of bandwidth observed in different cases. Black points indicate that the AS does not perform an attack, red when it performs a prefix hijacking attack against six different guard prefixes, and blue when the defense has been deployed. For the defense, we choose a threshold of 8 (out of 20 CPAs) and relay AS hop neighborhood of *(left:)* one ($k = 1$) and *(right:)* two ($k = 2$).

3. For neighbor-AS discovery (§6.3), the AS neighborhood of a relay changes rarely, especially relative to BGP churn. If relays repeat this test daily, it should be sufficient to detect AS neighborhood changes.

In the final piece of the defense, agreement between the CPAs on a given guard is required, but the individual CPAs may not have performed their tests at the same time. However, the agreement is only based on whether a potential attack is detected, not comparing network information between CPAs. While some CPAs may not detect an attack as early as others, this can be tuned by a bandwidth/detection delay tradeoff and is unrelated to BGP churn.

Defense precision. We now estimate the false positive rate of our distributed service—that is, we determine how likely it is to consider a guard as under attack when it is not.

We first note that the AS hop neighborhood technique (§6.2) has no false positives since, given sufficient time, a node should be able to completely explore its AS neighborhood.

To estimate the overall false positive rate, we obtained traceroutes using RIPE probes [31] between the top 50 client ASes and the top 50 guard ASes, by bandwidth. We then conducted traceroutes from the client ASes to the penultimate ASes to each of the guards, based on the aforementioned set of traceroutes. This corresponds to the subpath property check (see §6.1). We filtered out malformed traceroutes, such as those in which no routers along with path responded. From a practical perspective, these indicate that these CPAs are unsuitable and other CPAs would have to be chosen.

Assuming that no attacks are currently taking place, we compute the false positive rate as the fraction of guards that would be perceived as being under attack by our distributed detection system. We define the fraction of CPAs which detect a guard as under attack as the *flag rate* of that guard. For completeness, we present the raw flag rates in Appendix A. Our data shows that a good selection cutoff is 40%—that is, the Tor network should consider a guard to be under attack when at least 40% of the CPAs consider an attack to be taking place against that guard. Using this threshold of 40%, we obtain an overall false positive rate of 1/43 (2.32%). We note that the 40% threshold was used in our earlier experiments (see Figure 10) to assess the system's ability to detect actual attacks.

Improving the defense to mitigate the allergy attack. The defense, as described above, enables an *allergy attack* [4]: an adversary may cause DoS by conducting control-plane attacks *against a CPA*, gaining control of packets going towards it, including the return ICMP messages of its traceroutes. In the worst case, the adversary may cause arbitrary detection (or potentially prevent detection) of any set of guard nodes by forging ICMP responses.

We improve our system by having two groups of CPA nodes perform the ensemble defense on each other. We rename our group residing in client ASes as CPA-A, and place a second set of trusted nodes, CPA-B, in various guard ASes.

We describe the ensemble defense setup from the perspective of CPA-A, but the same applies to CPA-B due to symmetry. Our ensemble defense setup proceeds as follows.

We first need to find viable nodes in CPA-A. We perform the ensemble defense between CPA-A and CPA-B nodes to find the base flag rate of each node (number of nodes which detect this node as under attack). We then select nodes which have base flag rates lower than some threshold *selectionT* as the viable CPA-A nodes (we choose 30% for both groups). We also set a second threshold, *exclusionT* > *selectionT*, which defines when a usable node in CPA-A is no longer allowed to contribute to the ensemble defense, i.e., once it has been flagged by too many viable CPA-B nodes.

Our results (see Appendices A and B) show that setting a *selectionT* at 30% yields an overall false positive rate of ap-

proximately 12% (52% for CPA-B). Since we only select the viable nodes, the number of excluded nodes are not important as long as there are enough viable nodes. Our data also shows that the false positive rate for detecting guards/CPA-Bs when under attack, as given earlier in this section, is mostly unaffected when we apply this filtering (which shows that a single pass will find all viable nodes). We set *exclusionT* at 40%, as before, since our results show that this is effective in detecting the adversary.

In addition to viable CPA-A nodes running the ensemble defense on guard relays, viable CPA-A nodes and viable CPA-B nodes are running it on each other. When a CPA node from either group is flagged by enough nodes such that it crosses the *exclusionT* threshold, we exclude its probe results from future evaluations until it drops back below the threshold.

Now consider the case where the adversary attacks some set of viable CPA-A, CPA-B and guard relays. Each attacked node in CPA-A provides one vote against each node in CPA-B (and vice versa). Therefore, the attacker needs to conduct control-plane attacks against at least *exclusionT* - *selectionT* CPA-A nodes (10% in our setting) to affect nodes in CPA-B.

However, each attacked CPA-A node will be flagged by many benign CPA-B nodes. This follows from our previous results. Therefore these attacked CPA-A nodes will be excluded from the network after one detection cycle. Given sufficiently large sets of CPA-A/CPA-Bs, and that our adversary is not flagrantly attacking large numbers of ASes, this system is stable.

8 Discussion

Attacks at Tor exit relays. The paper so far explores the control-plane attacks that manipulate the routing of the traffic to Tor guard relays. In fact, similar attacks can be launched to attract traffic destined to Tor exit relays. Importantly, such attacks are sufficient to perform traffic correlation: while traditional correlation attacks require that the attacker observe and control traffic flows in the same direction (e.g., client-to-guard and exit-to-destination), Sun et al. [35] demonstrate that it is possible to exploit TCP ACKs to perform an attack when the attacker can observe and control traffic flows in different directions. Attacking a Tor exit relay might be more attractive for the attacker than attacking the destination, especially if the destination is a popular website which will have tremendous fallout in terms of bandwidth that the attack would have to handle and observable delays for users of that website. Our defenses can be set up symmetrically with CPAs probing the exits to detect an attacker performing this attack.

Targeted attacks. By attacking Tor guard and exit relays, an adversary would be able to de-anonymize *some* clients whose ingress and egress traffic are observable by the adversary. In some situations, the adversary may decide to attack a *targeted* client prefix, since, for example, it could be cheaper in terms of bandwidth or a high-profile client resides in that prefix. In this situation, the victim client cannot rely on distributed CPAs to help detect the attack, since the CPAs probe only guards or exits, but not clients whose identities are anonymous and not apriori known. We argue next that a variant of our defense techniques would still defend against this attack.

The victim client can perform a traceroute to its guard, and ask the guard do a traceroute to itself as well. The traceroute from the guard will now be routed through the adversary, while the one to the guard will not (the reverse of the case when the attacker targets the guard). The similar checks for AS-hop discrepancy, subpath property, and AS neighborhood would work just as before, except that it is done in the reverse direction (e.g., the guard, instead of the client, would now check the subpath property, and the AS neighborhood information is collected by the client instead of the guard).

Bandwidth considerations. We do not consider the bandwidth overhead of the attacker AS for launching the attack. We believe that, given that a knowledgeable adversary targets its attack mainly at guard nodes and potentially prefixes containing only transit routers, the bandwidth overhead from carrying non-Tor traffic can be minimized by careful selection of target prefixes. This is especially true for the longest prefix attacker which has more freedom in its selection of target prefixes.

Denial of service (DoS). Tor's intention to move to a single guard design with no rotations [7] appears to run contrary to our recommendation of changing one's guard if it is under attack. We however see this as a necessary tradeoff due to the attack (not the defense): the user must choose between (i) increasing its risk of compromise by using different guards [7, 19], (ii) risk using the guard which is under attack (i.e., ignore detected control-plane attacks), or (iii) not use Tor at all when his guard is under attack. The latter clearly presents a DoS opportunity, although it is unclear whether DoS is acceptable if the alternative is compromised anonymity. Since the second almost certainly exposes the user to a significant loss of anonymity, we suggest the first or third approaches as the least worst options.

9 Conclusion

This paper explores in-depth the implications to the Tor overlay network of attacks against its underlay (i.e., the Internet). We quantify the effect of previously proposed longest-prefix

hijacking attacks on Tor, finding that 92% of Tor's overall guard bandwidth is vulnerable to attack.

We additionally measure the effectiveness against Tor of control-plane manipulations based on false short path advertisements, and derive the conditions under which such attacks are possible. Our findings reveal that such attacks are fairly limited in scope (affecting only a few clients), but that the overall percentage increase in the adversary's visibility into guard traffic is significant when the attack is successful. We also show that the limited range of shortest path-based attacks makes them particularly difficult to detect.

To counter network-level threats to Tor, we introduce a suite of detection mechanisms that allow clients to detect attacks and avoid selecting guards that could result in compromises to their anonymity. Using accurate topological models of the Internet and probabilistic analyses based on real network traces, we explore in-depth the benefits, effectiveness, and limitations of our techniques. Finally, we introduce a distributed architecture for detecting longest-prefix hijacking attacks against Tor with high accuracy.

Although this paper focuses on control-plane attacks against Tor, our defenses may be more broadly applicable: our techniques rely on structural properties of the Internet and overlay nodes that can attest to their local views of the network. We posit that defenses analogous to those proposed may be useful for protecting other distributed systems (e.g., Bitcoin) against similar threats. We leave the exploration of this space as a future research direction.

Acknowledgments

We thank Paul Syverson for many helpful discussions about this work, and for the anonymous reviewers for their insightful comments and suggestions. We are also grateful to Paul Syverson and Aaron Johnson for pointing us towards important datasets and related work. This paper is partially funded from National Science Foundation grants CNS-1149832, CNS-1527401, CNS-1453392, and CNS-1513734. The findings and opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

References

[1] I. C. Avramopoulos and J. Rexford. Stealth Probing: Efficient Data-Plane Security for IP Routing. In *USENIX Annual Technical Conference (USENIX-ATC)*, 2006.

[2] CAIDA UCSD IPv4 Routed /24 Topology Dataset. http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml.

[3] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[4] S. P. Chung and A. K. Mok. Allergy Attack against Automatic Signature Generation. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2006.

[5] CIDR Report 18 Aug 2015. http://www.cidr-report.org/as2.0/.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, August 2004.

[7] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson. One Fast Guard for Life (or 9 Months). In *Privacy Enhancing Technologies Symposium (PETS)*, 2014.

[8] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Protocol Misidentification Made Easy with Format-Transforming Encryption. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[9] M. Edman and P. Syverson. AS-Awareness in Tor Path Selection. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[10] 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 *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2012.

[11] N. Feamster and R. Dingledine. Location Diversity in Anonymity Networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2004.

[12] D. Fifield. meek. https://trac.torproject.org/projects/tor/wiki/doc/meek.

[13] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, 2001.

[14] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking (ToN)*, 9(6):733–745, 2001.

[15] S. Hahn and K. Loesing. Privacy-preserving Ways to Estimate the Number of Tor Users. Technical Report 2010-11-001, Tor Project, November 2010.

[16] A. Houmansadr and N. Borisov. SWIRL: A Scalable Watermark to Detect Correlated Network Flows. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[17] A. Houmansadr, C. Brubaker, and V. Shmatikov. The Parrot is Dead: Observing Unobservable Network Communications. In *IEEE Symposium on Security and Privacy (Oakland)*, 2013.

[18] R. Jansen and N. Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[19] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users Get Routed: Traffic Correlation on Tor By Realistic Adversaries. In *ACM Conference on Computer and Communications Security (CCS)*, November 2013.

[20] A. M. Johnson, P. Syverson, R. Dingledine, and N. Mathewson. Trust-based Anonymous Communication: Adversary Models and Routing Algorithms. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[21] J. Juen. Protecting Anonymity in the Presence of Autonomous System and Internet Exchange Level Adversaries. Master's thesis, University of Illinois at Urbana-Champaign, 2012.

[22] D. Kedogan, D. Agrawal, and S. Penz. Limits of Anonymity in Open Environments. In *Information Hiding Workshop (IH)*, 2002.

[23] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4): 582–592, April 2000.

[24] M. Lepinski. BGPSec Protocol Specification. Draft draft-ietf-sidr-bgpsec-protocol-04, Internet Engineering Task Force, 2012.

[25] MaxMind's GeoIP Database. https://dev.maxmind.com/geoip/geoip2/geolite2/.

[26] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[27] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy (Oakland)*, 2005.

[28] V. N. Padmanabhan and D. R. Simon. Secure Traceroute to Detect Faulty or Malicious Routing. *ACM SIGCOMM Computer Communication Review*, 33(1):77–82, 2003.

[29] H. N. Phong, A. Yasuhito, and Y. Masatoshi. Anti-RAPTOR: Anti Routing Attack on Privacy for a Securer and Scalable Tor. In *IEEE International Conference on Advanced Communication Technology (ICACT)*, 2015.

[30] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, Internet Engineering Task Force, 2006.

[31] RIPE Atlas. https://atlas.ripe.net/.

[32] Routeviews Prefix to AS mappings Dataset for IPv4 and IPv6. http://www.caida.org/data/routing/routeviews-prefix2as.xml.

[33] RouteViews Project. http://www.routeviews.org/.

[34] Snakes on a Tor Exit Scanner. https://gitweb.torproject.org/torflow.git/tree/HEAD:/NetworkScanners/ExitAuthority.

[35] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: Routing Attacks on Privacy in Tor. In *USENIX Security Symposium (USENIX)*, Aug. 2015.

[36] The CAIDA AS Relationships Dataset, <2014-06-01>. http://www.caida.org/data/as-relationships/.

[37] Tor Flow. https://gitweb.torproject.org/torflow.git/.

[38] Tor Project, Inc. Tor Metrics Portal. https://metrics.torproject.org/.

[39] C. Wacek, H. Tan, K. Bauer, and M. Sherr. An Empirical Evaluation of Relay Selection in Tor. In *Network and Distributed System Security Symposium (NDSS)*, February 2013.

[40] T. Wan, E. Kranakis, and P. C. van Oorschot. Pretty Secure BGP, psBGP. In *Network and Distributed System Security Symposium (NDSS)*, 2005.

[41] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. StegoTorus: A Camouflage Proxy for the Tor Anonymity System. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[42] R. White. Architecture and Deployment Considerations for Secure Origin BGP (soBGP). Draft draft-white-sobgp-architecture-02, Internet Engineering Task Force, 2006.

[43] H. Yu, J. Rexford, and E. Felten. A Distributed Reputation Approach to Cooperative Internet Routing Protection. In *Workshop on Secure Network Protocols (NPSec)*, 2005.

[44] J. Zhang, J. Rexford, and J. Feigenbaum. Learning-based Anomaly Detection in BGP Updates. In *ACM SIGCOMM Workshop on Mining Metwork Data*, 2005.

[45] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao. Practical Defenses Against BGP Prefix Hijacking. In *ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2007.

[46] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis. A Light-weight Distributed Scheme for Detecting IP Prefix Hijacks in Real-time. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

A Detailed Results for False Positive Testing (for Guards)

Table 2 presents the raw results for determining the precision of our distributed service (see §7).

Each row represents the assessment of all of the CPAs for a particular guard. The second column in the table reports how many of the CPAs (out of 50) were able to conduct traceroutes to the guard. Of the CPAs that were able to conduct traceroutes, the rightmost column reports the false flag rate—that is, the fraction of those CPAs that incorrectly perceived that the guard was under attack. This table also applies to CPA-Bs, which reside in guard ASes.

B Detailed Results for False Positive Testing (for CPAs)

Table 3 presents the assessment of the CPAs, as perceived by CPA-Bs (nodes in guard ASes). Each row in the table represents the assessment of all of the guards for a particular CPA. The second column in the table reports how many of the CPA-Bs (out of 50) were able to conduct traceroutes to the CPA. Of the CPA-Bs that were able to conduct traceroutes, the rightmost column reports the false flag rate—that is, the fraction of those guards that incorrectly perceived that the CPA was under attack.

Guard #	# of CPA Assessments	FPR
1	38	0.052632
2	27	0.074074
3	42	0.095238
4	39	0.102564
5	39	0.102564
6	19	0.105263
7	36	0.111111
8	44	0.113636
9	26	0.115385
10	34	0.117647
11	40	0.125000
12	37	0.135135
13	44	0.136364
14	42	0.142857
15	41	0.146341
16	40	0.150000
17	43	0.162791
18	35	0.171429
19	35	0.171429
20	29	0.172414
21	38	0.184211
22	38	0.184211
23	36	0.194444
24	41	0.195122
25	41	0.195122
26	41	0.195122
27	40	0.200000
28	43	0.209302
29	38	0.210526
30	42	0.238095
31	29	0.241379
32	28	0.250000
33	36	0.250000
34	39	0.256410
35	39	0.256410
36	19	0.263158
37	43	0.279070
38	35	0.285714
39	43	0.348837
40	31	0.354839
41	41	0.365854
42	42	0.380952
43	38	0.684211

Table 2. Raw results for false positive testing. Here, the CPAs assess whether the guards are under attack. The rows (i.e., guard numbers) are sorted by increasing false positive rate.

CPA #	# of CPA-B Assessments	FPR
1	26	0.115385
2	41	0.121951
3	40	0.125000
4	39	0.128205
5	29	0.137931
6	43	0.139535
7	39	0.153846
8	39	0.153846
9	43	0.162791
10	24	0.166667
11	35	0.200000
12	36	0.222222
13	40	0.225000
14	40	0.225000
15	39	0.230769
16	38	0.236842
17	38	0.236842
18	25	0.240000
19	44	0.250000
20	32	0.250000
21	34	0.264706
22	41	0.268293
23	40	0.275000
24	42	0.285714
25	40	0.300000
26	43	0.302326
27	43	0.302326
28	33	0.303030
29	42	0.309524
30	41	0.317073
31	40	0.325000
32	43	0.325581
33	42	0.333333
34	33	0.363636
35	40	0.375000
36	42	0.380952
37	38	0.394737
38	41	0.414634
39	38	0.421053
40	40	0.425000
41	40	0.425000
42	42	0.428571
43	42	0.547619
44	42	0.642857
45	42	0.666667
46	43	0.883721

Table 3. Raw results for false positive testing. Here, the CPA-Bs assess whether the CPAs are under attack. The rows (i.e., CPA numbers) are sorted by increasing false positive rate.