Armon Barton* and Matthew Wright

DeNASA: Destination-Naive AS-Awareness in Anonymous Communications

Abstract: Prior approaches to AS-aware path selection in Tor do not consider node bandwidth or the other characteristics that Tor uses to ensure load balancing and quality of service. Further, since the AS path from the client's exit to her destination can only be inferred once the destination is known, the prior approaches may have problems constructing circuits in advance, which is important for Tor performance. In this paper, we propose and evaluate DeNASA, a new approach to AS-aware path selection that is *destination-naive*, in that it does not need to know the client's destination to pick paths, and that takes advantage of Tor's circuit selection algorithm. To this end, we first identify the most probable ASes to be traversed by Tor streams. We call this set of ASes the *Suspect AS* list and find that it consists of eight highest ranking Tier 1 ASes. Then, we test the accuracy of Qiu and Gao AS-level path inference on identifying the presence of these ASes in the path, and we show that inference accuracy is 90%. We develop an AS-aware algorithm called DeNASA that uses Qiu and Gao inference to avoid Suspect ASes. DeNASA reduces Tor stream vulnerability by 74%. We also show that DeNASA has performance similar to Tor. Due to the destination-naive property, time to first byte (TTFB) is close to Tor's, and due to leveraging Tor's bandwidth-weighted relay selection, time to last byte (TTLB) is also similar to Tor's.

Keywords: Anonymity, Tor

DOI 10.1515/popets-2016-0044
Received 2016-02-29; revised 2016-06-02; accepted 2016-06-02.

1 Introduction

Tor is a popular anonymous communication system that allows Internet users to access the web anonymously or circumvent censorship [10]. Past research has shown that end-to-end timing attacks [30] and traffic correlation attacks [24, 29] may be used to deanonymize Tor users by eavesdropping adversaries who simultaneously observe both traffic from a client to the Tor network and traffic from the Tor network to the client's requested destination. Thus, any autonomous systems (ASes) that are traversed on both of those sections of the path from the client to her destination would make ideal candidates for adversarial eavesdropping vantage points [11, 12, 19]. Adversaries that observe traffic from those vantage points are termed *AS-level adversaries*.

To evaluate the risks posed by AS-level adversaries, it is necessary to determine the set of ASes that appear on both of those sections of the path. One method is to infer the AS-level path, using *Qiu and Gao AS-level path inference* [27]. Another method is to infer the AS-level path using traceroute measurements [23]. Recently, Juen et al. found that Qiu and Gao AS-level path inference significantly overestimates the number of ASes traversed by Tor traffic; thus, this method directly is not suitable for predicting or mitigating the risk of AS-level adversaries to Tor [21].

All previously proposed AS-aware path selection methods [11, 20, 28] share a key disadvantage: the significant performance benefit from pre-constructing circuits is diminished due to needing to check the circuits for adversarial eavesdropping vantage points once the destination is known and possibly construct new circuits if no safe ones are available. Starov et al. show an increase in page load time by 41% in their AS-aware Tor client, *Astoria* [28]. They suggest that the performance loss in page load times are due to: 1) the inability to preemptively construct circuits to the same degree as Vanilla Tor, and 2) the time cost of computing AS-level paths and checking those paths for adversarial vantage points.

1.1 Contributions

In this paper, we first show empirical results suggesting that Qiu and Gao AS-level path inference, though unreliable for full paths, is sufficiently reliable in detecting

***Corresponding Author: Armon Barton:** UT Arlington, E-mail: armon.barton@mavs.uta.edu
Matthew Wright: UT Arlington, E-mail: mwright@uta.edu

high-ranking ASes [5]. We then propose a novel method for AS-awareness in Tor that leverages the reliable information from Qiu and Gao's approach and evaluate its anonymity and network performance.

In particular, we first identify the most probable adversarial eavesdropping vantage points, denoted as *Suspect ASes*, to be traversed by Tor streams. Then, we test the Qiu and Gao AS-level path inference method for its ability to infer the presence of each individual *Suspect AS* in the path and show that the mean accuracy is 90%. Based on this finding, we developed DeNASA, a novel AS-aware path selection algorithm that does not need to know the client's destination in advance and thereby can take advantage of the performance optimization achieved by having pre-constructed circuits that are fully ready to be used. We term this property as being *destination-naive*. We designed a TorPS [31] simulation in which 34.4% of Tor streams were vulnerable. This is slightly higher than what most previous works suggest [11, 12, 19, 20]. We show in the simulation that DeNASA reduces Tor stream vulnerability by 74%.

We also implement this AS-aware path selection algorithm in the Tor source code and test its network performance in Shadow. We show that TTFB is not significantly different than Vanilla Tor due to the destination-naive property. Using the same simulation we show that TTLB is not significantly different than Vanilla Tor.

2 Background

2.1 Tor

Tor is a circuit-based low-latency anonymous communication system designed to anonymize TCP-based applications [10]. Tor consists of approximately 7000 relays [15] distributed throughout the world making up an overlay network across the Internet. Each Tor relay is deployed voluntarily by people around the world who wish to support anonymous communications and can supply the bandwidth and computational power. These volunteer-run nodes accommodate millions of users each day [15]. Each user's Tor client selects a path (called a *circuit*) through the Tor network of exactly three Tor relays in which each relay in the path knows its predecessor and successor, but no other relays in the path. In the circuit, the first hop is called a *guard relay*, the second hop is called a *middle relay*, and the third hop is called an *exit relay*. The circuit-building design is sometimes referred to as onion routing, where the client negotiates session keys with each successive hop in the path incrementally until the final hop is reached where traffic proceeds from the final hop to the destination unencrypted. A client can connect to several different destination websites through one circuit due to the fact that each circuit can be shared by several TCP streams.

In Tor, steps are taken to ensure low-latency web browsing. During circuit construction, there are delays due to the use of public key cryptography and network latency. These delays are mitigated by allowing clients to construct circuits in advance of using them for communicating. The process for selecting relays attempts to balance traffic over the available router bandwidth in the Tor network. The bandwidth capacity of the routers is periodically delivered to each client via the *consensus*, a document containing information about Tor relays. Load balancing is then achieved by selecting each router in proportion to its bandwidth capacity.

To prevent various attacks such as the *predecessor attack* [26, 33, 35], the selective denial of service attack [4], and statistical profiling attacks, clients use a single guard node as the first hop for every circuit she makes. This reduces the risk that an attacker will control the first hop on some of a user's circuits. A new guard is chosen only if the presently selected guard ever becomes unavailable, or if a period of 60 days to 9 months is reached [9], thereby reducing the probability of rotating to a compromised guard [19].

End-to-end timing attacks are performed by adversaries who passively observe traffic on both ends of a Tor stream, and thereby correlate traffic in order to identify the client and her requested destination. The Tor design is not resilient to end-to-end timing attacks, nor does it claim to be. An attacker passively watching traffic patterns on both ends of a Tor circuit will be able to deanonymize the user with high probability [10].

2.2 Network Layer Routing

The Internet can be viewed as an aggregate of Autonomous Systems (ASes) that are linked together at the network layer by high-bandwidth lines and fast routers. Each AS is a LAN that is owned and operated by a single organization, such as a company, government agency, university, or Internet service provider. Between ASes, the routing protocol on the Internet is

BGP (Border Gateway Protocol), which is fundamentally a distance vector protocol. BGP routers adhere to routing policies which include political, security, and economic constraints. For example, a corporate AS in Israel might be unwilling to forward transit traffic originating from an AS in Iran. On the other hand, the same AS in Israel might want to receive packets originating from other parts of the world which transit through ASes in Iran. The routing policies at each BGP router may differ; however, each BGP router keeps track of the exact path used to a particular destination, and it tells all neighbors the exact path it used to reach that particular destination. When a BGP router needs to reach a destination, it asks all neighbors what path they took and selects the best path based on its routing policies.

2.3 AS-Level Path Inference

Inferring AS-level paths from arbitrary source and destination pairs across the Internet is essential in analyzing a path for AS-level adversaries. A popular method for inferring AS-level paths is to perform a traceroute from the source to the destination. However, reliably compiling and distributing a complete database of traceroute data between all clients, Tor routers, and destinations to all users would be prohibitively expensive. We require a different approach.

2.3.1 Qiu and Gao Path Inference

ASes form business relationships with each other based on a variety of political, security, and economic constraints. A model introduced by Gao[13] abstracts these relationships into three types: customer-to-provider, provider-to-customer, and peer-to-peer. A customer always provides monetary transfer to the provider in exchange for the provider providing bandwidth to the customer. In peer-to-peer relationships, the two ASes have agreed to exchange traffic between each other on a quid-pro-quo basis, allowing monetary savings on transit costs between them [7]. We can imagine the ASes organized in graph theoretic terms as a forest of trees, with customers as leaves at the bottom and higher-level providers as root nodes at the top. Peer-to-peer relationships are extra edges that link the branches in the tree and can link different trees, particularly between root nodes.

Mao et al. introduce a method to infer AS-level paths by using this model of relationships [22]. In this method, a path is considered valid if, for every AS providing transit, there is a payee that is the neighbor of the transit provider. An invalid path is one in which there is at least one transit provider not paid by a neighbor in the path [7]. Valid paths are denoted as possessing the *valley-free* property: considering the tree-based graph structure of inter-AS relationships, there should be no path that goes down towards the customer leaves and then goes back up towards the root nodes.

Qiu and Gao[27] proposed an improvement on AS-level path inference by exploiting known paths that appear in BGP routing tables and inferring AS paths based on this information. Their method is also extended to infer a set of potential paths between a given source and a given destination. They show that their algorithm achieves an average accuracy of 60% for exactly matching the entire path and 80% for matching the path length. Additionally, they offer an AS path inference service called Swordqiu where users are able to access the most up-to-date inference results[3].

2.4 Related Work

2.4.1 Studies of the Threat

The vulnerability of Tor streams due to the threat of AS-level adversaries has been evaluated by several researchers. In Table 1, we summarize the findings of prior works that used AS-path inference to determine the percentage of vulnerable streams in Tor.

In the Internet, traffic may be routed asymmetrically, such that the AS path from the client to the server (the *forward* path) is different than that for return traffic from the server back to the client (the *reverse* path). Most works assume that the attacker should either see forward traffic from the client to the guard together with forward traffic from the exit to the destination (*Forward* in Table 1), or he should see reverse traffic from the guard to the client together with reverse traffic from the destination to the exit (*Reverse* in Table 1). This makes traffic analysis easier so as to link the two ends of the stream. A third case, based on the same assumption, is that the attacker could succeed when seeing either the *Forward* case or the *Reverse* case (*F/R* in Table 1).

More recently, however, it has been shown that the attacker can also succeed when seeing forward traffic

Author	Forward	Reverse	F/R	Asym	1 Guard
Feamster and Dingledine [12]	17.7%	16.1%	NA	NA	No
Edmond and Syverson [11]	10.9%	11.1%	17.8%	NA	No
Wacek et al. [32]	NA	NA	27.4%	NA	No
Juen [20]	7.1%	7.2%	11.2%	NA	No
Starov et al. [28]	NA	NA	NA	37.0%	Yes
Juen et al. [21]	11.6%	12.1%	21.6%	NA	No
This Work	17.2%	17.9%	NA	34.4%	Yes

Table 1. Comparison of results on the rates of vulnerable streams. Traffic can be observed in the *Forward* direction, the *Reverse* direction, either direction but the same on both ends (*F/R*), and either direction including asymmetrically (*Asym*).

from the client to the guard and reverse traffic from the destination to the exit, or vice versa [29]. We call this the *asymmetric* case (*Asym* in Table 1), and it covers the broadest possible set of observations.

According to most results, 11% to 28% of *forward or reverse* paths of Tor streams were vulnerable. According to Starov et al., 37% of asymmetric paths were vulnerable. Some reasons for the high variance in past results are due to the variations in Tor network sizes, AS-path inference techniques, and adversary models. We discuss adversary models in section 2.4.1.

Somewhat differently from these works, Sun et al. present a new attack suite called Raptor that can be launched by ASes to compromise Tor users [29]. They introduce an asymmetric traffic analysis attack that allows an adversary to deanonymize users if the adversary is able to observe any direction of the traffic at both ends of the Tor path. Additionally, via BGP interception they demonstrate a correlation attack on the live Tor Network that takes advantage of asymmetric routing on the Internet.

Similar to several prior works [11, 20, 21], we use Qiu and Gao's method for AS-level path inference [27]. Comparably to the recent work of Starov et al., we assume that adversaries can perform asymmetric traffic analysis attacks to deanonymize users. Note that for all entries in Table 1, apart from Starov et al. and *This Work*, Tor was not configured to use the recently deployed policy of using a single entry guard.

2.4.2 Defenses

Edman and Syverson proposed and evaluated an AS-aware path selection algorithm in which computational overhead is avoided on the client side by using a snapshot of the Internet AS-topology [11]. Akhoondi et al. proposed a Tor client called LASTor that minimizes latency in paths by considering geographic locations of relays [1]. LASTor is designed to avoid paths in which an AS can observe both sides. Juen et al. proposed an AS/IX-aware algorithm that reduces Tor vulnerable streams to 5.3%-11% [20]. Starov et al. presented an AS-aware Tor client called Astoria which reduces the number of vulnerable streams to under 5.1% [28].

Among these defenses, one common disadvantage is that the significant performance benefit from preconstructing circuits is diminished due to the difficulty in preemptively constructing safe circuits, particularly ones that are fully ready to be used, and due to the cost of computing and checking AS-level paths on the client side. If no available circuits are safe, new ones should be constructed, substantially increasing the delay before sending any requests to the server.

2.5 AS-Level Adversary Model

Due to the asymmetric nature of Internet routing, the AS path from source to destination can be different from the AS path from destination to source. In our model, we assume that AS-level adversaries can exploit the asymmetric nature of Internet routing to compromise users by observing *any direction of traffic* at both ends of the Tor path [28, 29]. We assume that the adversary may control the routers in any AS in the Internet. Since we do not know which ASes might be compromised, we simply say that when an AS appears on both ends of a path between the client and the destination, the AS is *well-positioned* and the path is *vulnerable*. Minimizing the chance of a path vulnerability, so defined, also reduces the risk for a user that any single compromised AS could link her with her destinations.

We now define this model more precisely. The *forward path* is one in which data in the TCP connection travels from client to guard and from exit to destination. The set of ASes from the *client (c)* to *guard (g)* that are traversed on the forward path are denoted as $P_{c\to g}$. The set of ASes from the *exit (x)* to *destination (d)* that are traversed on the forward path are denoted as $P_{x\to d}$. If $P_{c\to g} \cap P_{x\to d} \neq \emptyset$, then we say that the forward path is *vulnerable*. Of course, the ASes in this intersection may not be malicious or compromised by an adversary. As we do not know, however, which ASes are malicious, we assume that any AS can qualify.

Similarly, the *reverse path* is one in which data in the TCP connection travels from destination to exit and from guard to client. The set of ASes in between guard to client that are traversed on the reverse path are denoted as $P_{g\to c}$. The set of ASes in between destination to exit that are traversed on the reverse path are denoted as $P_{d\to x}$. If $P_{g\to c} \cap P_{d\to x} \neq \emptyset$, then the reverse path is vulnerable.

The combined asymmetric path from client to guard is denoted as $P_{c\leftrightarrow g} = P_{c\to g} \cup P_{g\to c}$. The asymmetric path from exit to destination is denoted as $P_{x\leftrightarrow d} = P_{x\to d} \cup P_{d\to x}$.

The *asymmetric path* from the client to her destination is vulnerable if $P_{c\leftrightarrow g} \cap P_{x\leftrightarrow d} \neq \emptyset$.

3 AS-Level Adversaries

In this section, we simulate the Tor network by generating Tor streams using TorPS and analyze those streams with Qiu and Gao AS-level path inference. The simulation will serve as a benchmark for analyzing our AS-aware approach (Section 4).

3.1 Experimental setup

TorPS [31] – a Tor path selection simulator – uses historical data to recreate network conditions that Tor users experienced in the real world [19]. Under these conditions, circuits are created according to past network state, and streams are attached to these circuits according to user behavior. Path selection in TorPS is based on the code in Tor version 0.2.3.25.

In our experiment, we configure TorPS to use a single guard and allow a new guard to be selected only if the first guard is unavailable. We use the 176 top Alexa [2] destination websites from 108 ASes that range in genre from news, social media, search engines, blogs, e-commerce, banking, classifieds, etc.

CC	Mean daily users	CC	Mean daily users
US	16.47%	ES	3.86%
DE	9.32%	BR	3.71%
RU	7.21%	IT	3.59%
FR	6.27%	PL	2.47%
GB	4.42%	JP	2.36%

Table 2. Top 10 countries by directly connecting users [15]

We test six distinct user models: 1) *5-destination*, 2) *10-destination*, 3) *all-destinations*, 4) *5.1-distinct*, 5) *5.2-distinct*, and 6) *5.3-distinct*. The *5-destination* user model is one in which each client selects five destinations uniformly at random from the set of 176 sites at the start of the simulation. During the simulation, the client then requests destinations uniformly at random from this pre-selected set. Similarly, the *10-destination* user model specifies that each client selects from among 10 pre-selected destinations for each connection. The *all-destination* user model is one in which all clients uniformly visit all 176 destinations.

We have three user models, *5.1-distinct*, *5.2-distinct*, and *5.3-distinct*, in which all users visit the same five destinations. The goal of using these models is to show the extent to which our findings depend on the user's choice of destinations. The five destinations for each model were pre-selected uniformly at random from the full set of destinations.

The client model is based on the top 10 countries by directly connecting users according to Tor Metrics [15]. Table 2 shows this weighted distribution of clients according to country. We simulate up to 10 clients from distinct ASes for each country totaling 92 clients from 92 distinct ASes. Client ASes are chosen partly from the list proposed by Edmond and Syverson [11], partly from the list proposed by Juen [20], and partly from CAIDA Top Ranking ASes [5].

We parsed a Tor Consensus from December 2014 and found that there were 353 ASes for all guard nodes and 454 ASes for all exit nodes. In our experiments, we simulate 49% of the Tor guard nodes from 329 ASes, and 43% of the Tor exit nodes from 380 ASes. Thus, 93% of

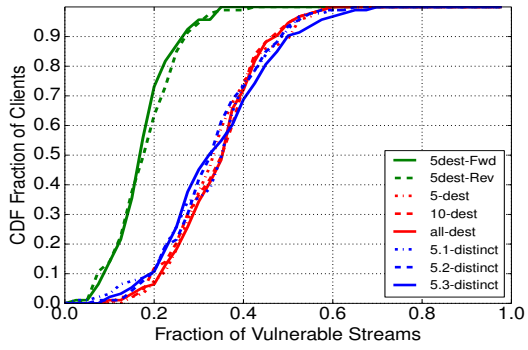

Fig. 1. Probability of a Tor client building a vulnerable stream using the default Tor path selection algorithm.

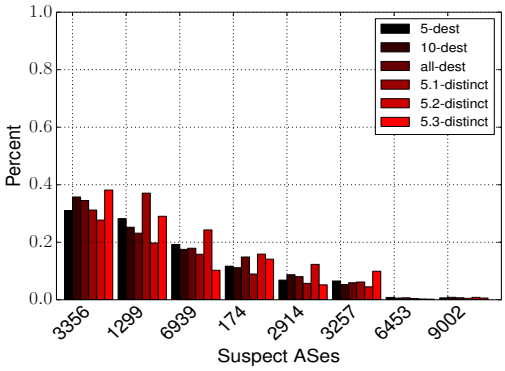

Fig. 2. Probability of Suspect AS being able to observe both sides of a Tor stream, given that the Tor stream was vulnerable.

Tor guard ASes and 81% of Tor exit ASes are represented in our simulations. Each client in the simulation builds 10,000 streams, and we thus build more than 900,000 streams per simulation. After all streams are generated, we use them as input to Swordqiu, the Qiu and Gao AS-level path inference tool. The *forward*, *reverse*, and *asymmetric* paths are analyzed and flagged as vulnerable according to the adversary model described in Section 2.5

3.2 Results

The probability of a Tor client building a vulnerable stream using the default Tor path selection algorithm is shown in Figure 1. The mean *forward path* and mean *reverse path* vulnerability rates for all clients using the *5-dest* user model was 17.2% and 17.9% respectively. The distribution of these two data sets are not statistically different. This suggests that, for all clients, the probability of building a vulnerable stream is statistically the same in either direction. Hence, targeting one direction over the other for AS-awareness does not give us any added advantage.

The mean *asymmetric path* vulnerable stream rate for all clients using the *5-destination* user model was 34.4%. The best and worst case vulnerable stream rate for a client was approximately 12% and 60% respectively. For 70% of clients, between 22% and 41% of their streams were vulnerable. A perhaps surprising result was that all user models, including the three *5.x-distinct* models in which all clients visited the same five destinations, produced statistically similar vulnerable streams for all clients. This suggests that clients may not evade the threat posed by well-positioned ASes simply by reducing the number of requested destinations nor by varying their destination distribution. The reason for this is the similar exit relay distribution for all circuits over time. This is because, whether going to different destinations or the same destination, the locations of the ever-changing exit relays provide a sufficient sample of ASes to get typical vulnerable stream rates. This notion, which we discuss further in Section 4, leads to our observation that certain ASes are predictably likely to be traversed from certain Tor exit nodes regardless of the destination. In the remaining TorPS experiments, we use the *5-destination* user model.

3.3 Suspect ASes

We observed a relatively small set of ASes that were consistently well-positioned in the generated Tor streams. We term this set of ASes as the *Suspect ASes*. From among the vulnerable streams in our experiments, we compute the percentage of times that each AS appeared on both ends of the stream, i.e. that each AS was in position to compromise the user's anonymity. Note that more than one AS may appear on both ends of a given stream. The results in Figure 2 show that the *Suspect AS* set does not significantly change for different user models. The top two ASes, AS3356 (Level 3 Communications) and AS1299 (TeliaSonera International Carrier), accounted for 27% to 38% and 20% to 37% of the vulnerable streams, respectively. Another 27% to 61% of vulnerable streams were due to the set of six other ASes shown in Figure 2. In the three *5.x-distinct* user models, we do see that some of the *Suspect ASes* are more or less common for different sets of destinations. The set of the top six ASes, however, remains the same for all user models.

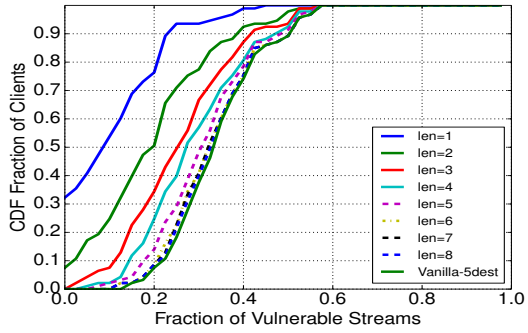

Fig. 3. Probability of a Tor client building a vulnerable stream under various adversary models that assume different number of compromised Suspect ASes. The *Vanilla-5dest* data set assumes any AS may be compromised.

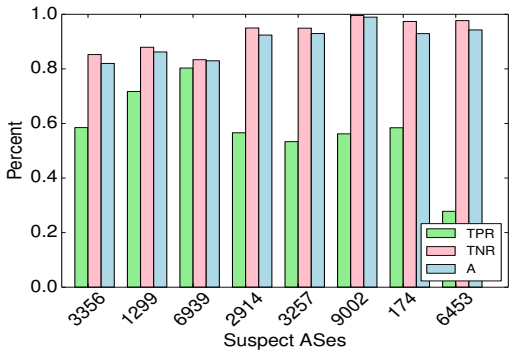

Fig. 4. True Positive, True Negative, and Accuracy rates of Qiu and Gao AS-level path inference and its ability to predict Suspect ASes in CAIDA traceroute paths.

Next, we evaluate the significance of the Suspect AS set. We augment the data from the simulation by tuning the adversary model and specifying that the adversary may only compromise Suspect ASes. By increasing the *length* of the Suspect AS set, we show in Figure 3 how the fraction of vulnerable streams increases. We observe that when the adversary controls a Suspect AS set of *len=1*(3356) and *len=2*(3356 and 1299), 32% and 9% of clients build zero vulnerable streams, respectively. As the adversary's resources increase from *len=1* to *len=6*, the mean vulnerable stream rate increases from 10.7% to 33.4%. However, when increasing the Suspect AS set length from *len=6* to *len=8*, vulnerable streams increased by only 0.4%. When it is assumed that *any* AS may be compromised by the adversary as described in our adversary model (Section 2.5), the mean vulnerable stream rate increased by only 0.6% to 34.4%. These results indicate that adversaries have little to gain by targeting ASes that are outside the Suspect AS set.

The eight Suspect ASes shown in Figure 2 will be used as input to our AS-aware approach described in Section 4.

3.4 Qiu and Gao Accuracy

We now evaluate the accuracy of Qiu and Gao path inference and its ability to predict whether or not a Suspect AS is in a path for arbitrary source-destination pairs. The accuracy of Qiu and Gao AS-level path inference with regards to assessing Tor has been recently questioned by Juen et al. [21]. They find that only 20% of inferred paths match paths acquired from traceroute measurements and that Qiu and Gao path inference significantly overestimates the number of ASes traversed by Tor traffic. They conclude that Qiu and Gao inference is not suitable for predicting or mitigating the risk of AS adversaries to Tor.

We note, however, that a path inference algorithm like the Qiu and Gao approach may still be accurate for identifying individual ASes on a path and for the eight members of our Suspect AS set in particular. In this case, an algorithm for AS-aware path selection that relies only on predicting the presence of these key ASes may be viable based on Qiu and Gao inference.

To determine the accuracy of Qiu and Gao in determining the presence or absence of the Suspect ASes, we set up an experiment where Qiu and Gao path inference is tested using a CAIDA traceroute data set [6] as ground truth. We are only concerned with how Qiu and Gao inference performs in predicting whether or not a *Suspect AS* is in the traceroute path. A Positive (P) case is one in which the AS is actually in the traceroute path, and a Negative (N) case is one in which the AS is not. A True Positive (TP) case is one in which Qiu and Gao inference correctly identifies that the AS is in the traceroute path. A True Negative (TN) case is one in which Qiu and Gao inference correctly identify that the AS is not in the traceroute path. In Figure 4, we present the results of three metrics of Qiu and Gao inference with respect to finding Suspect ASes:

$$\text{True Positive Rate } (TPR) = TP/P$$

$$\text{True Negative Rate } (TNR) = TN/N$$

$$\text{Accuracy } (A) = (TP + TN)/(P + N)$$

We find that Qiu and Gao inference has a mean accuracy of 90% taken over the eight Suspect ASes. We note that the TPR is quite low in some cases while the accuracy is high. This is due to the fact that Negative cases occurred 100 to 1000 times more frequently than Positive cases, causing the P/N ratio to be small.

We conclude that, with this accuracy, Qiu and Gao inference is suitable for predicting and mitigating AS-level adversaries in our simulation. On the other hand, our approach is designed in such a way that traceroute measurements or Qiu and Gao inference, or a combination of both, can be used in deployment.

4 AS-Awareness

We now introduce DeNASA, a new AS-aware approach to Tor path selection that is *destination-naive*, in that it allows clients to pre-build circuits that are fully ready to be used without knowing the requested destination in advance.

First, let us define T_{sus}, probability table that is input to the algorithm. T_{sus} is a table with rows equal to the number of *Exit ASes* and columns equal to the number of *Suspect ASes*. An *Exit AS* is an AS that a Tor exit relay resides in – denoted as AS^{ex}. In T_{sus}, each value ($PR_{i,j}$) represents the probability that a *Suspect AS* AS_i^{sus} will be traversed between an *Exit AS* AS_j^{ex} and possible destinations. T_{sus} values are obtained by inferring the AS-level path from all *Exit ASes* to all *Destinations* in our simulation.

In particular, ($PR_{i,j}$) is calculated as follows: Let AS_i^{sus} denote the i^{th} *Suspect AS*, AS_j^{ex} denote the j^{th} *Exit AS*, $Dest_k$ denote the k^{th} Destination, and let q be the total number of Destinations. Then $PR_{i,j}$ is equal to the total number of paths where AS_i^{sus} appears from AS_j^{ex} to $Dest_k$ for all $1 \leq k \leq q$, divided by the total number of paths from AS_j^{ex} to $Dest_k$ for all $1 \leq k \leq q$.

In a realistic deployment, it would not be practical to perform traceroute measurements from Internet destinations such as popular Web servers to Tor exit relays. As such, we obtain T_{sus} from forward paths from exit to destination, and not reverse paths from destination to exit. This means not having access to reverse path measurements if traceroute is used for AS path inference. Instead, we obtain T_{sus} from Qiu and Gao inference or by performing traceroute measurements from Tor exit relays to destinations across the Internet.

```
Data: P_{c↔g}
Data: Suspect AS list
Data: T_sus
Data: τ (a tunable parameter)
flag = 1;
select guard from Guard List;
while flag do
    select exit via Tor algorithm;
    foreach AS^sus in P_{c↔g} do
        if PR_{AS^sus,AS^ex} < τ then
            flag = 0;
        else
            flag = 1;
            break;
        end
    end
end
```

Algorithm 1: *e-select*, AS-aware path selection.

4.1 AS-Aware Path Selection Algorithm

Two methods make up the DeNASA path selection algorithm. The first method is called *e-select* (see Algorithm 1), and it determines how a client selects the exit relay. The data inputs are $p_{c\leftrightarrow g}$, *Suspect AS list*, T_{sus}, and τ (a tunable threshold). First, the client selects a guard node from the guard list. Then, via bandwidth weighting, the client selects an exit node. For all *Suspect ASes* that appear in the path from client to guard, if the value $PR_{AS,Exit}$ is less than τ, the chosen exit is accepted. If not, the algorithm selects and tests more exits until a suitable one is found. τ can be tuned from 0.0 (maximum security) to 1.0 (maximum flexibility).

The second method, called *g-select*, determines how a client adds guards to the guard list. In this method, a client first selects a potential guard via the Tor guard selection algorithm and then ensures that there is not a *Suspect AS* in the asymmetric path from client to guard. The client cycles through all live guards until one is found that meets this constraint. If there are no guards that meet the constraint, the client will select a guard as per the standard Tor guard selection algorithm. For g-select, the Suspect AS list *length* is reduced to *len=2* (3356 and 1299) in order to not introduce a guard placement attack (discussed in section 4.4).

Methods 1 and 2 can be switched on or off independently of each other. When both methods are switched on, we call the algorithm *g&e-select*.

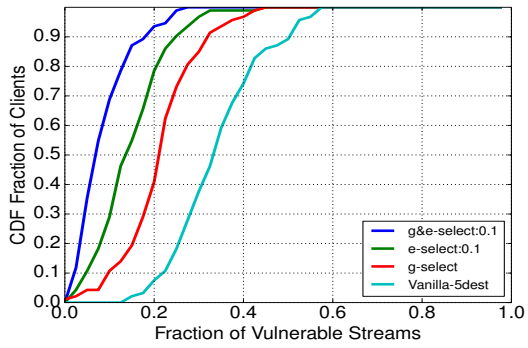

Fig. 5. Empirical CDF of the Fraction of Vulnerable Streams for e-select: $\tau = 0.1$, g-select, and g&e-select $\tau = 0.1$ compared to Vanilla Tor.

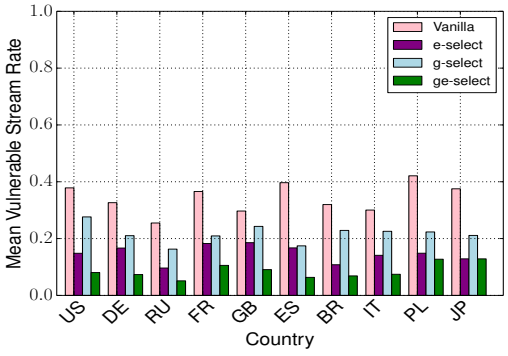

Fig. 6. Mean vulnerable stream rate for the top 10 countries by directly connecting users for e-select: $\tau = 0.1$, g-select, and g&e-select $\tau = 0.1$ compared to Vanilla Tor.

4.2 AS-Awareness Evaluation

To evaluate DeNASA's performance with respect to anonymity, we use the simulation described in Section 3 but with the new path selection algorithms implemented in TorPS. In the simulation, we generated over 900,000 streams for each curve. These streams were used as input to Qiu and Gao inference, and streams were flagged as being vulnerable based on the adversary model described in Section 2.5.

For *e-select:*, *g-select*, and *g&e-select:*, $T_{suspectTable}$ was trained using a subset of 92 destinations located in 55 ASes. Then, AS-awareness was tested using the full set of 176 destinations located in 108 ASes. This simulates clients requesting destinations that are both inside and outside of our training set.

The results can be seen in Figure 5 and are shown along with our results for Vanilla Tor. The *e-select* method performed best when τ was set to 0.1; this method reduced the mean vulnerable stream rate to 14.7%. The mean vulnerable stream rate for *g-select* was 21.7%. Finally, *g&e-select* with $\tau = 0.1$ had the best performance with a mean vulnerable stream rate of 8.8% compared to Vanilla Tor – a 74% decrease.

4.3 Client Location Diversity

To explore whether clients in different locations would have different levels of security in our scheme, we augment the data to reveal the performance of the AS-aware methods with respect to the client's location by country.

The mean vulnerable stream rate for clients in the countries in the top 10 by directly connecting users to Tor is shown in Figure 6. We observe that clients from all countries benefit from using any AS-aware method compared to Vanilla Tor; clients from all countries benefit more by using e-select over g-select; and clients from all countries apart from Poland and Japan benefit the most by using g&e-select over all other methods. Clients from Poland and Japan benefit the same by using g-select or g&e-select.

Clients from all countries realized an 81% to 85% decrease in mean vulnerable stream rate from Vanilla Tor compared to g&e-select.

4.4 G-select Evaluation

The g-select method may introduce a threat if: 1) a large set of clients can only access a few guards, or 2) a large set of guards are strictly accessible from few network locations. The first threat would give adversaries more opportunities to deploy relays that are positioned on both sides of Tor streams. The second threat would let the adversary gain information about the client's network location with high probability, based on the client's choice of guard. In this section we address both of these concerns. We also discuss how the results impact load distribution on guard nodes with g-select in use.

4.4.1 Guard Placement Attack

If clients have access to few guards that satisfy the g-select constraints, the g-select method may introduce a new attack we call the *guard placement attack*. A guard

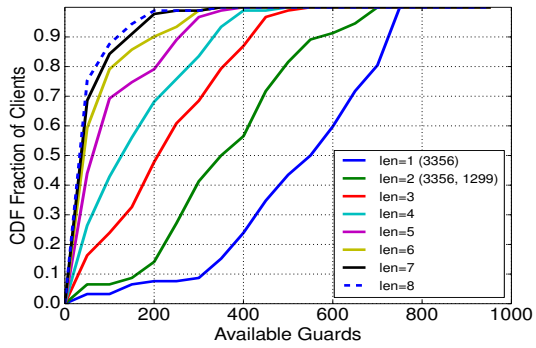

Fig. 7. Empirical CDF of available guards from 771 simulated guard nodes for clients using g-select with varying Suspect AS list length.

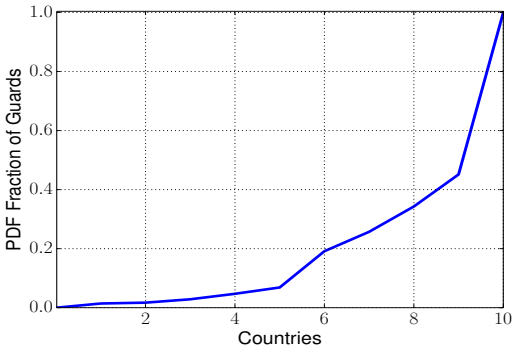

Fig. 8. Empirical PDF of fraction of Guards that are reachable form client connecting countries when g-select is in use.

placement attack is a relay-level attack [19] in which an attacker places a guard in a location that satisfies the g-select constraint for a set of clients who otherwise have few other guards to choose from. This causes the clients to select the attacker's guard with high probability and thereby harms their anonymity. This attack can be mitigated by relaxing the Suspect AS list length for g-select. When g-select is in use, we denote all guards that satisfy the g-select constraints and become available to clients as *available guards*. Figure 7 shows the available guards out of the 771 simulated guard nodes for clients using g-select. In each case, the Suspect AS list length was relaxed. When the full Suspect AS list was used (*len=8*), 75% of clients had access to at most 51 guards. In other words, an adversary launching a guard placement attack in this scenario would be competing with at most 51 guards to potentially compromise 75% of clients.

On the other hand, when the Suspect AS list length was relaxed to *len=2*, 94% of clients had access to between 101 to 695 guards, and 6% of clients had access to at most 100 guards. In this case, an adversary would be competing with at most 100 guards to potentially compromise 6% of clients. If the adversary intends to potentially compromise a larger percentage of clients, the number of guards to compete with grows linearly, increasing the cost of the attack. Similarly, when the Suspect AS list length was relaxed to *len=1*, 94% of clients had access to between 154 to 749 guards, and 6% of clients had access to at most 153 guards.

Based on these results, we believe that the longest Suspect AS list with an allowable risk from the guard placement attack is *len=2*. Therefore, a Suspect AS list of *len=2* (3356 and 1299) for the g-select method is used in all simulations in this study.

CC	Prob.	CC	Prob.
GB	14%	FR	9%
JP	14%	DE	9%
IT	12%	ES	8%
BR	11%	PL	7%
RU	10%	US	6%

Table 3. Distribution of origin countries from which clients may access guards when using g-select.

4.4.2 Client's Guard Choice

The guard placement attack would present an additional threat to clients if an adversary could determine a client's network location with high probability based on the client's guard choice. To address this concern, we analyzed reachability of guards from different countries when using g-select. As shown in Figure 8, 55% of guards were reachable from all 10 countries, and an additional 42% of guards were reachable from four to nine countries. However, 3% of guards were reachable from just one to three countries. If an adversary deployed a guard within this 3%, for example, the adversary could gain information about the client's country of origin with high probability. On the other hand, this attack is limited to a small subset of guards. Moreover, In Section 6.1.2, we introduce some suggestions that we believe would decrease the threat presented by the guard placement attack.

For all guards in our simulation, the distribution of origin countries from which clients may access guards when using g-select is shown in Table 3. Clients from GB and JP were represented approximately twice as much as clients from US and PL. For the majority of guards, the distribution indicates that an adversary should not be able to gain information about a client's origin country with high probability.

In future work, a similar study can be done to evaluate how much information an adversary can gain about a client's AS-level network location based on the client's guard choice when using g-select.

Note that the guard placement attack was evaluated with respect to the top ten client connecting countries in Tor, representing approximately 60% of directly connecting (non-Bridge) Tor users [15]. In future work, an evaluation can be done in which significantly more countries are included.

4.4.3 Guard Load Distribution

The results in this section indicate that, when g-select is in use: 1) most clients have access to a large set of guard nodes, and 2) most guard nodes are reachable from several client connecting countries. Hence, there should be no added congestion from g-select due to imbalanced guard load distribution. Additionally, in Section 5 we show that network performance of g-select supports our assertion that no significant congestion is added to the Tor Network when g-select is in use.

4.5 Shifting Suspects

The rate at which Tor streams were vulnerable for Vanilla, g-select, e-select, and g&e-select was 0.34, 0.22, 0.15, and 0.09 respectively. In Figure 9, we show the *conditional* probabilities – before and after DeNASA is applied – of a Suspect AS being able to observe both sides of a Tor stream, *given that the Tor stream was vulnerable*. The results indicate that g-select and g&e-select are properly avoiding AS3356 and AS1299 as expected. Additionally, the probability of being positioned on both ends is shifted from AS3356 and AS1299 to other Suspect ASes after DeNASA is applied. However, in these results, there is no indication that the probability of being well-positioned significantly increases for ASes that are outside the Suspect AS set. Hence, we believe that DeNASA does not cause ASes that are outside the Suspect AS set to be in a better position to attack by shifting a significant amount of Tor connections through those ASes.

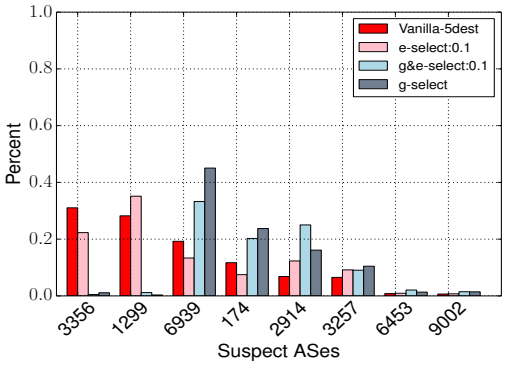

Fig. 9. Conditional probabilities – before and after DeNASA is applied – of Suspect ASes being able to observe both sides of a Tor stream given that the Tor stream was vulnerable.

A *strategic adversary* is one who knows DeNASA is being applied to circuit selection, and thereby targets Suspect ASes that have a high probability of being well-positioned after DeNASA is applied. For g-select and g&e-select, the conditional probability of being well-positioned increased dramatically for some Suspect ASes. Due to the rate at which Tor streams were vulnerable for g-select, Figure 9 shows that g-select users may not gain any additional advantage against a strategic adversary who specifically targets other Suspect ASes. On the other hand, due to the rate at which Tor streams were vulnerable for g&e-select, Figure 9 shows that g&e-select users do in fact gain a significant advantage against strategic adversaries who target other Suspect ASes.

4.6 AS-path lengths

Selecting a guard such that the path from client to guard does not contain AS3356 or AS1299 may cause clients to select guards that are geographically near compared to vanilla guard selection. This may be indicated by shorter AS-path lengths from client to guard. Selecting guards that are near to the client has some advantages. Sun et al. suggest that the risk of asymmetric traffic analysis due to BGP churn is mitigated when guards are selected that are closer to the client [29]. This subtopic of how g-select affects AS-path length is left for future work.

In the next section, we evaluate network performance of our proposed methods. We discuss more about deployment issues for DeNASA in Section 6.

5 Network Performance

To evaluate the network performance of Tor when using our AS-aware path selection algorithm, we simulate the Tor network in Shadow, a large-scale discrete-event Tor simulator. Shadow enables us to efficiently run accurate Tor Network experiments by modeling live network characteristics such as relay contributed bandwidth, bandwidth distribution from entry, middle, and exit relays, and geographic distributions of relays [16, 18]. Shadow runs real Tor software that allows researchers to experiment with modifications to Tor by directly modifying the Tor source code. Client performance in Shadow simulations is statistically similar to performance achieved through the live Tor network due to the use of the live Tor consensus.

The destination-naive property of DeNASA allows for statistically similar TTFB values compared to vanilla Tor because circuits can be pre-built without knowing the requested destination in advance, as described in Section 4. Additionally, relays are chosen with bandwidth weighted selection as a first priority during circuit construction. Thus, TTLB performance in DeNASA should be similar to vanilla Tor.

5.1 Tor Model

The Shadow configuration that we used consists of: 500 clients from the top 10 countries by directly connecting users shown in Table 2 [15]; 370 Tor relays sampled from the December 2014 consensus; and 70 destination servers from the Alexa list of top websites [2]. Table 4 shows the distribution of guard locations for the top 10 most common guard locations from our Tor model.

All clients and relays were assigned – based on their AS – to an enhanced network topology of 17,250 vertices and 150 million edges [17]. In this model, there are two classes of clients, *web clients* and *perf clients*. *Web clients* choose destination servers uniformly at random and then perform HTTP GET requests to download 320 KiB files from the modeled Tor network [16]. Each client measures the time from when the first request is made to

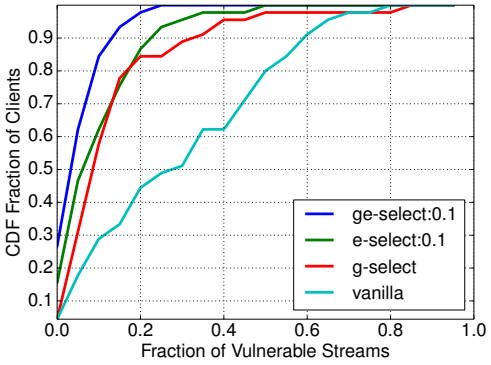

Fig. 10. Empirical CDF of the Fraction of Vulnerable Streams for e-select: $\tau = 0.1$, g-select, and g&e-select $\tau = 0.1$ compared to Vanilla Tor for streams generated in Shadow simulations.

CC	Prob.	CC	Prob.
DE	42%	GR	7%
US	23%	NL	6%
FR	20%	GB	4%
RU	8%	UA	4%
CZ	8%	SE	4%

Table 4. Distribution of guard locations for the top 10 most common guard locations from the Shadow simulation.

when the first byte is received ($TTFB$) and last byte is received ($TTLB$). Relays are selected by clients during circuit construction using bandwidth weights published in the consensus and according to one of the following path selection algorithms: vanilla, e-select, g-select, or g&e-select. *Perf clients* download 50 KiB files through the modeled Tor network. We validate our Tor model against live Tor by comparing the results of *perf clients* to historical Tor data from Tor Metrics [15].

We verify that the AS-aware methods are modeled correctly in Tor by generating streams from the shadow simulations similar to what was done with TorPS in Section 4. The results can be seen in Figure 10. The mean vulnerable stream rate for vanilla path selection, g-select, e-select:$\tau = 0.1$, and g&e-select:$\tau = 0.1$ were 34%, 16%, 15%, and 7% respectively. The results from the Shadow simulations were similar to the TorPS simulations and are shown in Table 5. The fundamental differences between the two simulations were: 1) the Tor

Simulator	g&e-select:0.1	e-select	g-select	vanilla
TorPS	8.8%	14.7%	21.7%	34.4%
Shadow	7%	15%	16%	34%

Table 5. Vulnerable Stream Comparison for TorPS vs. Shadow

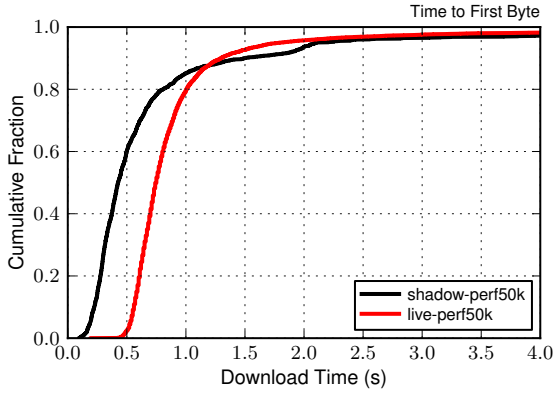

Fig. 11a. Time to first byte values for Shadow *perf clients* compared to live Tor perf [15] for fixed file size downloads of 50Kib.

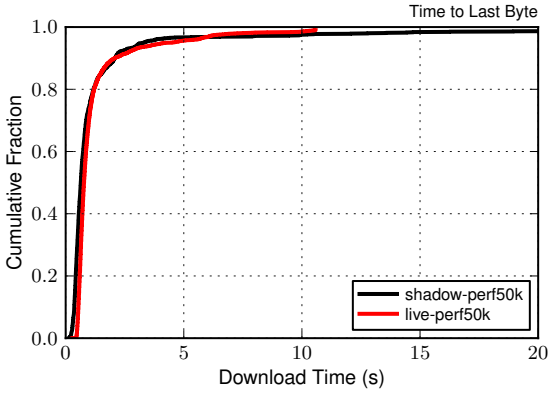

Fig. 11d. Time to last byte values for Shadow *perf clients* compared to live Tor perf [15] for fixed file size downloads of 50Kib.

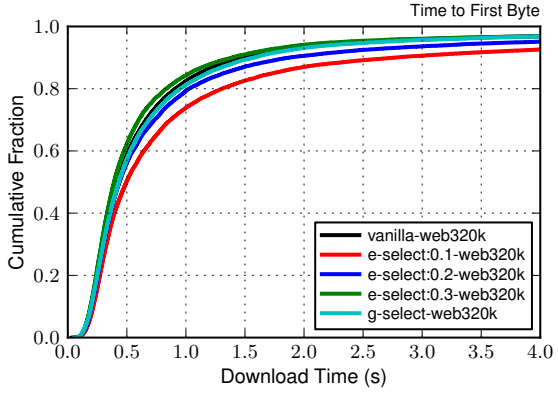

Fig. 11b. Time to first byte values for *web clients* when using g-select, e-select:0.1, e-select:0.2, and e-select:0.3 compared to vanilla Tor.

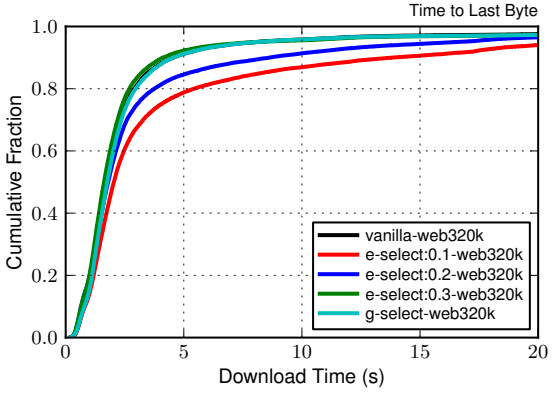

Fig. 11e. Time to last byte values for *web clients* when using g-select, e-select:0.1, e-select:0.2, and e-select:0.3 compared to vanilla Tor.

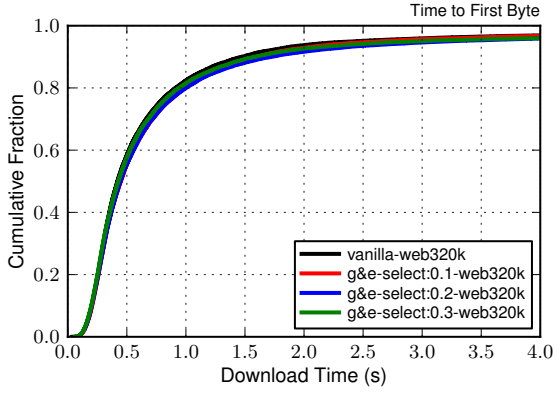

Fig. 11c. Time to first byte values for *web clients* when using g&e-select:0.1, g&e-select:0.2, and g&e-select:0.3 compared to vanilla Tor.

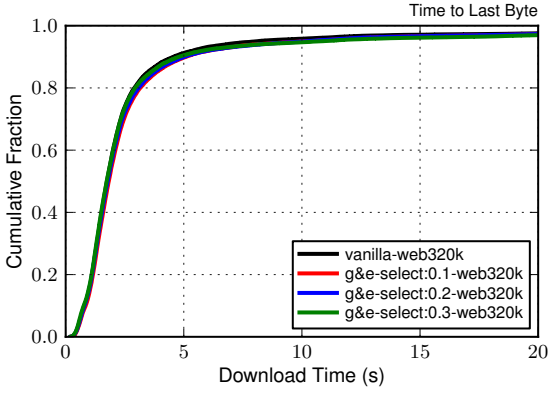

Fig. 11f. Time to last byte values for *web clients* when using g&e-select:0.1, g&e-select:0.2, and g&e-select:0.3 compared to vanilla Tor.

network size was significantly smaller for the Shadow simulations, and 2) the number of streams generated from the Shadow simulations were significantly less.

5.2 Performance Results

In Figures 11a and 11d, we plot live Tor performance for fixed file size downloads of 50 Kib from historical Tor network data from Tor Metrics [15] during the period of December 2014. The live performance can be compared to Shadow *perf clients* to validate that client performance in our model was statistically similar to that of live Tor. We tuned the client-to-relay ratio of our simulation so that TTLB would be very similar for Shadow and live Tor, as we see in Fig. 11d. For these settings, however, TTFB is slower in live Tor than our Shadow simulations (Fig. 11a).

Figures 11b and 11e show TTFB and TTLB values for g-select, e-select:0.1, e-select:0.2, and e-select:0.3 compared to vanilla Tor. On the most secure setting for e-select ($\tau = 0.1$), 80% of downloads were from approximately 0.0s to 0.5s slower in TTFB and 0.0s to 2.5s slower in TTLB compared to vanilla Tor. As τ is increased to 0.3, the difference in TTFB and TTLB for e-select compared to vanilla Tor approach zero. These results indicate that exit nodes are being slightly overloaded when e-select constraints are set high ($\tau = 0.1$). However, as the e-select constraints are relaxed, the exit node selection becomes diverse enough such that no significant congestion is added to the Network.

The TTFB and TTLB values for g-select were similar to vanilla Tor. These results and results from Figure 7 indicate that guard nodes are not being overloaded when g-select is in use.

Figures 11c and 11f show TTFB and TTLB values for g&e-select:0.1, g&e-select:0.2, and g&e-select:0.3 compared to vanilla Tor. The TTFB and TTLB performance for each g&e-select method was very similar to vanilla Tor. When g-select is combined with e-select, the exit node selection policy is relaxed due to the guarantee that AS3356 and AS1299 are not in the path from client to guard. This allows clients to select exit nodes that would otherwise not satisfy the e-select constraints. This leads to some exit nodes being not as loaded when g&e-select is in use compared to e-select, and thus g&e-select:0.1 outperforms e-select:0.1 in TTFB and TTLB.

6 Discussion and Future Work

In this section we discuss some deployment ideas for g-select and e-select, and caveats within the DeNASA approach. We additionally suggest ideas for future work.

6.1 G-select

The g-select method requires less modification to Tor compared to e-select. In this method, the client would perform AS-path inference once every two to nine months (when a client chooses a new guard) or when the current guard becomes unavailable. In our current version, the client would choose a guard in which AS3356 and AS1299 are not exist in the path from the client to the guard. Deploying g-select does not require any modification to Tor relays or directory authorities because all the work is done on the client side.

6.1.1 Mobile clients

The frequency that the g-select method may be called from the client side will increase if the client is mobile. A mobile client will be connecting from different ASes over time and thus must re-evaluate the path between client to guard more often. Moreover, the g-select method may cause mobile clients to add more guards to their guard list compared to stationary clients. This may cause mobile clients to be more vulnerable to relay-level adversaries who control a percentage of deployed guard nodes [19].

On the other hand, if a client only makes small movements to adjacent ASes within their geographic area, the AS path from client to guard may not change enough to cause the client to select a new guard. How client mobility affects guard selection when using the g-select method is an open question for future work.

Since relay-level attacks are effective [19] and probably cheaper than AS-level attacks to deploy, we would suggest reducing the chance of success for a relay-level attacker by having g-select not trigger a new guard selection when a mobile client connects from different ASes until the effects of client mobility have been further studied in future work. Furthermore, future work would include an analysis showing DeNASA's resilience against relay-level adversaries in general.

6.1.2 Constraints

In g-select, we needed to pick a trade-off in the length of the Suspect AS list, where longer lists prevented more AS-level attacks but led to increased vulnerability to guard placement attacks. The trade-off is not the same for every client, however, as some clients have more available guards even with a longer list. It might be possible for clients to measure their own susceptibility to guard placement attacks and then choose an appropriate setting for the list length. This would allow less restricted clients to have greater protection against AS-level adversaries, with minimal additional risk of guard placement attacks, while more restricted clients could use a list of length zero if needed.

For g-select, the Suspect AS list combination (AS3356 and AS1299) was selected based on the results shown in Figure 7. The data shown accounts for eight out of 28 possible combinations of the Suspect AS list, for $len = 2$. It might be possible to tune the Suspect AS list combinations to allow restricted clients to have greater protection against the guard placement attack.

The g-select method introduces constraints which determine how clients select guards. The constraints change based on client's location. The constraints may also vary if g-select is modified such that random combinations of the Suspect AS list are used for each client. These characteristics, or other modifications to g-select, may cause g-select to naturally induce Tor guard nodes into non-disjoint guard-sets [9, 14]. Future work would include an analysis of how g-select may induce guard-sets and what the implications may be for anonymity.

6.2 E-select

We now discuss deployment of the e-select method.

The client side requirements for e-select are: the AS path from client to guard (discussed in Section 6.1), the Suspect AS list (found in Section 3 to be the eight highest ranking Tier 1 ASes, but subject to change over time), and T_{sus}. Thus, the deployment challenge in the e-select method is in obtaining T_{sus}, and obtaining the Suspect AS list. For T_{sus} we propose two methods. The first method is designed around using Qiu and Gao AS-path inference, and the second is designed around using traceroute measurements for inferring AS-paths.

The Qiu and Gao deployment requires the directory authorities to perform AS-path inference from all exit relays to a number of AS destinations. The number of AS destinations will be determined by how often T_{sus} is recalculated. For example, if the directory authorities are configured to deliver a new T_{sus} to the clients every week, then the number of AS destinations will be determined by how many AS-paths that can be inferred in this amount of time.

With a larger destination set size, T_{sus} values should be more accurate. Though it is not feasible to infer AS-paths for all possible AS destinations, our results show that AS-awareness can be achieved for a larger set of AS destinations than what was originally inferred for calculating T_{sus}. In Section 4, we calculated T_{sus} for a subset of 55 AS destinations and tested AS-awareness for the full set of 108 AS destinations. This simulated clients requesting destinations that are both inside and outside the training set. On the other hand, traceroute measurements can be performed from all exit nodes to a number of AS destinations. The results can be aggregated at the directory authorities where T_{sus} would be calculated and delivered to all clients. In this method, only forward paths from exit to destination are realistically attainable. The accuracy may be enhanced if Qiu and Gao inference is used to acquire the reverse paths from destination to exit in combination with traceroute measured paths. We calculated T_{sus} from forward paths only in order to simulate the constraint of not having access to reverse path measurements.

One disadvantage of using traceroute measurements from exit relays is that compromised exit relays may spoof AS paths if traceroute measurements are used from exit to destination to obtain T_{sus}. In particular, the compromised exit could remove any suspect ASes and thereby increase its probability of being selected by Tor clients using e-select. Assuring reliable traceroutes initiated from a potentially unreliable host is a challenging problem to be sure.

6.3 Suspect AS List

The Suspect AS list is expected to change as the underlying AS topology of the Internet changes and as the Tor Network scales. As such, we suggest that the Suspect AS list should be computed and updated by the directory authorities and sent to clients along with the T_{sus}. The Suspect AS list can be computed by running a simulation at the directory authorities similar to the TorPS simulation in Section 3.

6.4 Destination Set

As discussed in Section 3.1, 176 destinations were used from the top websites according to Alexa [2] that range in genre from news, social media, search engines, blogs, e-commerce, banking, classifieds, and more.

Originally, 200 destinations were selected. For 24 destinations, however, the AS paths from some exits to those destinations, or from those destinations to some exits were not resolvable with the AS path inference method. For this reason, we removed these 24 destinations from the destination set.

In a realistic deployment of DeNASA, destinations could be chosen based on actual AS destinations that are requested by Tor users.

6.5 Improvements

The DeNASA method could be enhanced to evade IX adversaries. In this deployment, the set of Suspect ASes would additionally include sets of peering ASes that peer at Internet Exchange Points. T_{sus} would additionally include probabilities that a TCP connection would traverse a set of peering ASes from an exit AS. The sets of peering ASes would additionally be considered via T_{sus} during circuit creation for e-select, and during guard selection for g-select. The effects of an enhanced DeNASA method to evade IX adversaries is left for future work.

7 Conclusions

Our work is in response to the call for more sophisticated path selection algorithms in anonymous communications which resist the prevalent threat of AS-level adversaries. As such, we have demonstrated the effectiveness of our destination-naive AS-aware scheme in terms of anonymity and network performance. First, we identified the most probable ASes to be traversed by Tor streams. We call this set of ASes the *Suspect AS* set and found that it consists of eight highest ranking Tier 1 ASes. Then, we validated the accuracy of Qiu and Gao AS-level path inference on identifying the individual ASes from the eight members of our Suspect AS set. We developed an AS-aware algorithm that reduces Tor stream vulnerability by 74% simply by avoiding Suspect ASes. We demonstrate that DeNASA has performance similar to Tor. Due to the destination-naive property, time to first byte is close to Tor's, and due to leveraging Tor's bandwidth-weighted relay selection, time to last byte is also similar to Tor's.

8 Acknowledgments

We would like to thank the reviewers for providing critical comments throughout several revisions of this paper. In particular, we thank our shepherds Roger Dingledine and Paul Syverson, as well as the other anonymous reviewers for introducing the guard placement attack and providing insight that greatly helped improve the paper. This work was supported in part by NSF awards number CNS-1423163 and CNS-0954133.

References

[1] Masoud Akhoondi, Chu Yu, and Harsha V Madhyastha. LASTor: A low-latency AS-aware Tor client. In *IEEE S&P*, 2012.

[2] Alexa.com. Alexa top sites., June 2015. http://www.alexa.com/topsites.

[3] bgpVista. Swordqiu, March 2015. http://www.bgpvista.com/asinfer.php.

[4] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *CCS*, 2007.

[5] CAIDA. CAIDA AS ranking, June 2015. http://as-rank.caida.org/.

[6] CAIDA. The CAIDA UCSD IPv4 routed /24 topology dataset, June 2015. http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml.

[7] CAIDA. The CAIDA AS relationships, January 2016. http://www.caida.org/data/as-relationships/.

[8] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *PETS*, 2003.

[9] Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson. One fast guard for life (or 9 months). In *HotPETs*, 2014.

[10] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

[11] Matthew Edman and Paul Syverson. AS-awareness in Tor path selection. In *CCS*, 2009.

[12] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *WPES*, 2004.

[13] Lixin Gao. On inferring autonomous system relationships in the Internet. *ACM/IEEE Transactions on Networks (TON)*, 9(6), 2001.

[14] Jamie Hayes and George Danezis. Guard sets for onion routing. In *PETS*, 2015.

[15] Tor Project Inc. Tor Metrics, June 2015. https://metrics.torproject.org.

[16] Rob Jansen, Kevin S Bauer, Nicholas Hopper, and Roger Dingledine. Methodically modeling the Tor network. In *CSET*, 2012.

[17] Rob Jansen, John Geddes, Chris Wacek, Micah Sherr, and Paul Syverson. Never been KIST: Tor's congestion management blossoms with kernel-informed socket transport. In *USENIX Security*, 2014.

[18] Rob Jansen and Nicholas Hopper. Shadow: Running Tor in a box for accurate and efficient experimentation. In *NDSS*, 2012.

[19] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *CCS*, 2013.

[20] Joshua Juen. Protecting anonymity in the presence of autonomous system and Internet exchange level adversaries. Master's thesis, University of Illinois, http://hdl.handle.net/2142/34363, 2012.

[21] Joshua Juen, Aaron Johnson, Anupam Das, Nikita Borisov, and Matthew Caesar. Defending Tor from network adversaries: A case study of network path prediction. In *PETS*, 2015.

[22] Z Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. On AS-level path inference. In *SIGMETRICS*, 2005.

[23] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H Katz. Towards an accurate AS-level traceroute tool. In *SIGCOMM*, 2003.

[24] Steven J Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *IEEE S&P*, 2005.

[25] Steven J Murdoch and Piotr Zieliński. Sampled traffic analysis by Internet-exchange-level adversaries. In *PETS*, 2007.

[26] Lasse Overlier and Paul Syverson. Locating hidden servers. In *IEEE S&P*, 2006.

[27] Jian Qiu and Lixin Gao. Cam04-4: AS path inference by exploiting known AS paths. In *GLOBECOM*, 2006.

[28] Oleksii Starov, Rishab Nithyanand, Adva Zair, Phillipa Gill, and Michael Schapira. Measuring and mitigating AS-level adversaries against Tor. In *NDSS*, 2016.

[29] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: Routing attacks on privacy in Tor. In *USENIX Security*, 2015.

[30] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, 2001.

[31] TorPS. TorPS: The Tor path simulator. http://torps.github.io.

[32] Chris Wacek, Henry Tan, Kevin S Bauer, and Micah Sherr. An empirical evaluation of relay selection in Tor. In *NDSS*, 2013.

[33] Matthew Wright, Micah Adler, Brian N Levine, and Clay Shields. Defending anonymous communications against passive logging attacks. In *IEEE S&P*, 2003.

[34] Matthew K Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *TISSEC*, 7(4), 2004.

[35] Matthew K Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Passive-logging attacks against anonymous communications systems. *TISSEC*, 11(2), 2008.