

How Secure are Secure Interdomain Routing Protocols?

Full version from February 23, 2010

Sharon Goldberg
Microsoft Research

Michael Schapira
Yale & UC Berkeley

Peter Hummon
Princeton

Jennifer Rexford
Princeton

ABSTRACT

In response to high-profile Internet outages, BGP security variants have been proposed to prevent the propagation of bogus routing information. To inform discussions of which of these variants should be deployed in the Internet, we *quantify* the ability of the main protocols (origin authentication, soBGP, S-BGP, and data-plane verification) to blunt traffic-attraction attacks; *i.e.*, an attacker that deliberately attracts traffic to drop, tamper, or eavesdrop on packets.

Intuition suggests that an attacker can maximize the amount of traffic he attracts by *widely* announcing a *short* path that is not flagged as bogus by the secure protocol. Through simulations on an empirically-determined AS-level topology, we show that this strategy is surprisingly effective, even when the network uses an advanced security solution like S-BGP or data-plane verification. Worse yet, we show that these results *underestimate* the severity of attacks. We prove that finding the most damaging strategy is NP-hard, and show how counterintuitive strategies, like announcing longer paths, announcing to fewer neighbors, or triggering BGP loop-detection, can be used to attract even more traffic the strategy above. These counterintuitive examples are not merely hypothetical; we searched the empirical AS topology to identify specific ASes that can launch them. Finally, we find that a clever export policy can often attract almost as much traffic as a bogus path announcement. Thus, our work implies that mechanisms that police export policies (*e.g.*, defensive filtering) are crucial, even if S-BGP is fully deployed.

1. INTRODUCTION

The Internet is notoriously vulnerable to *traffic attraction* attacks, where Autonomous Systems (ASes) manipulate BGP to attract traffic to, or through, their networks. Attracting extra traffic enables the AS to increase revenue from customers, or drop, tamper, or snoop on the packets [1, 2, 3]. While the proposed extensions to BGP prevent many attacks (see [4] for a survey), even these secure protocols are susceptible to a *strategic* manipulator who deliberately exploits their weaknesses to attract large volumes of traffic to its network. Given the difficulty of upgrading the Internet to a new secure routing protocol, it is crucial to un-

derstand how well these protocols blunt the impact of traffic attraction attacks.

1.1 Quantifying the impact of attacks.

We evaluate the four major extensions to BGP, ordered from weakest to strongest: origin authentication [5], soBGP [6], S-BGP [7], and data-plane verification [4, 8]. While the stronger protocols prevent a *strictly* larger set of attacks than the weaker ones, these security gains often come with significant implementation and deployment costs. To inform discussions about which of these secure protocols should be deployed, we would like to *quantitatively* compare their ability to limit traffic attraction attacks. Thus, we simulate attacks on each protocol on an empirically-measured AS-level topology [9, 10, 11], and determine the percentage of ASes that forward traffic to the manipulator.

Performing a quantitative comparison requires some care. It does *not* suffice to say that one protocol, say S-BGP, is four times as effective as another protocol, say origin authentication, at preventing a *specific type of attack strategy*; there may be *other attack strategies* for which the quantitative gap between the two protocols is significantly smaller. Since these more clever attack strategies can just as easily occur in the wild, our comparison must be in terms of the *worst possible attack* that the manipulator could launch on each protocol. To do this, we put ourselves in the mind of the manipulator, and look for the *optimal* strategy he can use to attract traffic from *as many ASes as possible*.

However, before we can even begin thinking about optimal strategies for traffic attraction, we first need a model for the way traffic flows in the Internet. In practice, this depends on complex local routing policies used by each AS, which are not publicly known. However, the BGP decision process breaks ties by selecting shorter routes over longer ones, and it is widely believed [12] that policies depend heavily on economic considerations. Thus, conventional wisdom and prior work [13, 12, 14] suggests basing routing policies on business relationships and AS-path lengths. While this model (used in many other studies, *e.g.*, [1, 15]) does *not* capture all the intricacies of interdomain routing, it is still very

useful for *gaining insight* into traffic attraction attacks. All of our results are attained within this model.

1.2 Thinking like a manipulator.

If routing policies are based on AS path lengths, then intuition suggests that it is optimal for the manipulator to announce the *shortest path* that the protocol does not reject as bogus, to *as many neighbors* as possible. Depending on the security protocol, this means announcing a direct connection to the victim IP prefix, a fake edge to the legitimate destination AS, a short path that exists but was never advertised, a short path that the manipulator learned but is not using, or even a legitimate path that deviates from normal export policy. Indeed, we use simulations on a measured AS-level topology to show that this “smart” attack strategy is quite effective, even against advanced secure routing protocols like S-BGP and data-plane verification.

Worse yet, we show that our simulations *underestimate* the amount of damage manipulator could cause. Through counterexamples, show that the “smart” attack is surprisingly *not optimal*. In fact, the following bizarre strategies can sometimes attract even more traffic than the “smart” attack: announcing a *longer* path, exporting a route to *fewer* neighbors, or triggering BGP’s *loop-detection* mechanism. In fact, we show that prefix hijacking (i.e., originating a prefix you do not own) is *not* always the most effective attack against today’s BGP! These counterexamples are not merely hypothetical—we identify specific ASes in the measured AS-level topology that could launch them. Moreover, we prove that it is NP-hard to find the manipulator’s optimal attack, suggesting that a comprehensive comparison across protocols must remain elusive.

1.3 Our findings and recommendations.

While we necessarily *underestimate* the amount of damage a manipulator could cause, we can make a number of concrete statements. Our main finding is that secure routing protocols only deal with one half of the problem: while they do restrict the *paths* the manipulator can announce, they fail to restrict his *export policies*. Thus, our simulations did show that, when compared to as BGP and origin authentication, soBGP significantly limits the manipulator’s ability to attract traffic by announcing bogus short paths to all its neighbors. However, even in a network with advanced security solutions like S-BGP or even data-plane verification, we found that a manipulator can still attract large volumes of traffic by cleverly *manipulating his export policies*. Indeed, we found that announcing a *short* path is often less important than exporting that path to the *right set of neighbors*. Thus:

- Advanced security protocols like S-BGP and data-plane verification do *not* significantly outperform

soBGP for the “smart” attacks we evaluated.

- Defensive filtering of paths exported by stubs provides a level of protection that is *at least* comparable to that provided soBGP, S-BGP and even data-plane verification.
- Tier 2 ASes are in the position to attract the largest volumes of traffic, even in the presence data-plane verification and defensive filtering (of stubs).
- *Interception attacks* [2, 1]—where the manipulator both attracts traffic and delivers it to the destination—are easy for many ASes, especially large ones.

We could quibble about whether or not manipulating export policies even constitutes an *attack*; after all, each AS has the right to decide where it announces paths. However, our results indicate that a clever export policy can attract almost as much traffic as a bogus path announcement. Indeed, Section 6.1 presents an example where an AS in the measured topology gains almost as much exporting a provider-learned path to another provider, as he would by a prefix hijack (announcing that he owns the IP prefix). Thus, our results suggest that addressing traffic attraction attacks requires both mechanisms that prevent bogus path announcements (e.g., soBGP or S-BGP) as well as mechanisms that police export policies (e.g., defensive filtering).

1.4 Roadmap.

Section 2 presents the routing model, threat model, and our experimental setup. Section 3 describes the vulnerabilities of the secure routing protocols and presents an example of how a manipulator can attract traffic by exploiting them. Section 4 describes and evaluates the “smart” *attraction* attacks, and Section 5 uses both theory and simulation to analyze *interception* attacks. Section 6 presents counterexamples, found in real network data, that prove that the “smart” attacks are not optimal. Section 7 shows that finding the optimal attack strategy is NP hard. Section 8 presents related work, and Section 9 discusses the effect of our modeling assumptions on our results and provides further recommendations.

This full version also contains a variety of supplementary information in the appendix. Appendix A has our treatment of sibling relationships, and Appendix B has the details of our simulation methodology. Appendix D presents a counterexample from Section 6 in more detail, and Appendix E presents a supplementary example that shows how an interception attack can fail, and correct an error in [1]. Proofs of our theorems are in Appendices F-G. Finally, Appendix H presents versions of all the graphs in the body of this paper, computed from a different AS topology datasets [10, 11] than the graphs in the body [9], showing the remarkable agreement in the trends we report.

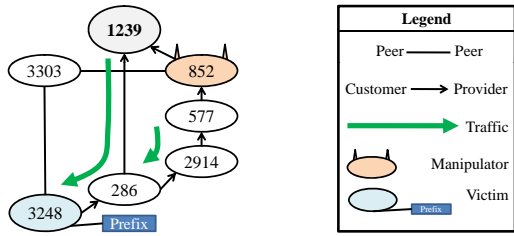


Figure 1: Subgraph from CAIDA’s AS Graph.

2. MODEL AND METHODOLOGY

We first present a model of interdomain routing and routing policies, based on the standard models in [16] and the Gao-Rexford conditions [13], followed by our threat model for traffic attraction, and finally overview our experimental setup.

2.1 Modeling interdomain routing.

The AS graph. The interdomain-routing system is modeled with a labeled graph called an *AS graph*, as in Figure 1. Each AS is modeled as a single node and denoted by its AS number. Edges represent direct physical communication links between ASes. Adjacent ASes are called *neighbors*. Since changes in topology typically occur on a much longer timescale than the execution of the protocol, we follow [16] and assume the AS-graph topology is static. BGP computes paths to each destination IP prefix separately, so we assume that there is a unique *destination IP prefix* to which all other nodes attempt to establish a path. As shown in Figure 1, there is a single AS (AS 3248) that rightfully ‘owns’ the destination IP prefix under consideration.

Establishing paths. In BGP, an AS first chooses an outgoing edge on which it forwards traffic based on a local ranking on outgoing paths, and then announces this path to some subset of its neighbors. To model this, we assume that each node n has a set of *routing policies*, consisting of (a) a *ranking* on outgoing paths from n to the destination d , and (b) a set of *export policies*, a mapping of each path P to the set of neighbors to which n is willing to announce the path P . We say that node n has an *available path* aPd if n ’s neighbor a announced the path “ aPd ” to n . If an available path aPd is ranked higher than the outgoing path that node n is currently using, then a *normal* node n will (a) forward traffic to node a , and (b) announces the path $naPd$ to all his neighbors as specified by his export policies.

Business relationships. We annotate the AS graph with the standard model for bilateral business relationships in the Internet [13]; while more complicated business relationships may exist in practice, the following is widely believed to capture the majority of the economic relationships. As shown in Figure 1, there are two kinds of edges: *customer-provider* edges (where the customer pays the provider for connectivity, repre-

sented with an arrow from customer to provider), and *peer-to-peer* edges (where two ASes owned by different organizations agree to transit each other’s traffic at no cost, represented with an undirected edge). Because we some of our results are based on CAIDA’s AS graph [9], we also consider *sibling-to-sibling* edges. Details about our treatment of siblings is in Appendix A. Finally, our theoretical results sometimes use an assumption from [13] that captures the idea that an AS cannot be its own customer:

GR1 AS graph contains no customer-provider cycles.

2.2 Modeling routing policies.

In practice, the local routing policies used by each AS in the Internet are arbitrary and not publicly known. However, because we want to understand how false routing information propagates through the Internet, we need to concretely model routing policies. Since it is widely believed that business relationships play a large role in determining the routing policies of a given AS [12, 13], and we have reasonably accurate empirical maps of the business relationships between ASes [9, 10, 11], these relationships will form the basis of our model.

Rankings. BGP is first and foremost designed to prevent loops. Thus, we assume that node a rejects an announcement from its neighbor b if it contains a *loop*; if node a appears on the path that node b announces. Beyond that, we can think of the process ASes use to select routes as follows; first applying local preferences, then choosing shortest AS paths, and finally applying a tie break. Since the local preferences of each AS are unknown, and are widely believed to be based (mostly) on business relationships, we model the three step process as follows:

LP Local Preference. Prefer outgoing paths where the next hop is a customer over outgoing paths where the next hop is a peer over paths where the next hop is a provider.

SP Shortest Paths. Among the paths with the highest local preference, chose the shortest ones.

TB Tie Break. If there are multiple such paths, choose the one who’s next hop has lowest AS number.¹

Our model of local preferences is based on on Gao-Rexford condition **GR3**, and captures the idea that an AS has an economic incentive to prefer forwarding traffic via customer (that pays him) over a peer (where no money is exchanged) over a provider (that he must pay). Notice that this implies that can sometimes prefer

¹We need a consistent way to break ties. In practice, this is done using the geographic distance between routers and router IDs. Since our model does not incorporate geographic distance or individual routers, we use AS number instead.

a *longer* path! (e.g., Figure 1, AS 852 prefers the five-hop customer path through AS 577 over the four-hop provider path through AS 1239.)

Export Policies. Our model of export policies is based on the Gao-Rexford condition **GR2**:

GR2 AS b will only announce a path via AS c to AS a if at least one of a and c are customers of b .

GR2 captures the idea that an AS should only willing to load his own network with transit traffic if he gets paid to do so. However, because **GR2** does *not* fully specify the export policies of every AS (for instance, an AS could decide to export paths to only a *subset* of his customers), it does not suffice for our purposes. Thus, we model normal export policies as follows:

NE An AS will announce *all* paths to *all* neighbors *except* when **GR2** forbids him to do so.

2.3 Threat model.

One strategic manipulator. We assume that all ASes in the AS graph behave *normally*, i.e., according to the policies in Section 2.1 - 2.2, except for a *single manipulator* (e.g., AS 852 in Figure 1). We leave models dealing with colluding ASes for future work.

Normal ASes and normal paths. We assume that every *normal* AS uses the routing policies Section 2.2; thus, the *normal path* is the path an AS (even the manipulator) would chose if he used the normal rankings of Section 2.2, and *normal export* is defined analogously. (e.g., In Figure 1, the manipulator AS 852’s *normal path* is through his customer AS 577.) We shall assume that every normal AS knows its business relationship with his neighbors, and also knows the next hop it chooses for forwarding traffic to a given destination. In order to evaluate the effectiveness of each secure routing protocol, we assume that ASes believe everything they hear, *except* when the secure routing protocol tells them otherwise. As such, we do not assume that ASes use auxiliary information to detect attacks, including knowledge of the network topology or business relationships between distant ASes *etc.*, unless the secure routing protocol *specifically* provides this information.

Attraction v.s. Interception attacks. In an *attraction attack*, the manipulator’s goal is to attract traffic, i.e., to convince the maximum number of ASes in the graph to forward traffic that is destined to the *victim IP prefix* via the manipulator’s own network.² To model the idea that a manipulator may want to eavesdrop or tamper with traffic before forwarding it on to

²We acknowledge that a manipulator may want to attract traffic from only a *specific subset* of ASes. However, since we lack empirical data to quantify that subset of ASes that a given manipulator may want to attract, in this paper we assume that the manipulator would like to *maximize* the volume of traffic that he attracts.

the legitimate destination, we also consider *interception attacks*. In an interception attack, the manipulator has the additional goal of ensuring that he has an *available path to the victim*. This is in contrast to a attraction attack, where the manipulator is allowed, but not required, to *create a blackhole* where he has no working path to the victim IP prefix (e.g., Figure 8).

Attack strategies. To capture the idea that the manipulator is strategic, we allow him to be more clever than the normal ASes; specifically, we allow him to use knowledge of the global AS graph and its business relationships in order to launch his attacks. (However, the strategies we consider usually require only knowledge this available locally at each AS.) An attack strategy is a set of routing announcements and forwarding choices that deviates from the normal routing policies specified in Section 2.2. An attack strategy may include, but is not limited to:

- Announcing an unavailable or non-existent path.
- Announcing different paths to different neighbors.
- Announcing an legitimate available path that is *different* from the *normal path*.
- Exporting a path (even the legitimate normal path) to a neighbor to which *no path* should be announced to according to the normal export policies.

Indeed, one might argue that some of these strategies do not constitute ‘dishonest behavior’. However, it is important to consider these strategies in our study, since we shall find that they can sometimes be used to attract as much traffic as the traditional ‘dishonest’ strategies (e.g., announcing non-existent paths).

Attacks outside our model. This paper focus on traffic attraction attacks; we do not consider other routing security issues, for instance, mismatches between the control- and data-plane [3, 8], or traffic deflection attacks, where a manipulator wants to divert traffic from himself or some distant, innocent AS [4].

2.4 Experiments on empirical AS graphs.

All the results and examples we present are based empirically obtained snapshots of the Internet’s AS graph annotated with business relationships between ASes.

Algorithmic simulations. At the core of our experiments is our *routing tree algorithm* (presented in Appendix B.1) that determines the paths that each AS uses to reach the destination prefix under the assumption that each AS ‘normally’ uses the routing policies of Section 2.2. Because we run a large number of experiments over the full (30K node) AS graph, we avoid the heavy message-passing approach used by standard BGP simulators; instead, we use lightweight algorithmic approach based on breadth-first search. The routing tree algorithm also is also used to simulating the

result of a manipulator’s attack strategy. As discussed in Appendix B.1, we simulate a bogus path announcements by ‘seeding’ the routing tree algorithm with the bogus path, and simulate strategic export policies by removing certain links between the manipulator and his neighbors.

Average case analysis. Since the influence of an attack strategy depends heavily on the locations of the manipulator and the victim in the AS graph, we run simulations across many (manipulator, victim) pairs. Rather than reporting average results, we plot the *distribution* of the fraction of ASes that direct traffic to the manipulator. We by no means believe that a manipulator would select its victim at random; however, reporting distributions allows us to measure the extent to which a secure protocol can blunt the power of the manipulator, determine the fraction of victims that a manipulator could effectively target, and identify positions in the network that are effective launching points for attacks. Ideally, to determine how damaging a given attack strategy can be, we would have liked to run simulations over *every* (manipulator, victim) pair in the AS graph. However, this would require $(30K)^2$ simulations per dataset, which would be prohibitive. Instead, we run experiments a randomly-chosen (manipulator, victim) pairs. We found that running 60K experiments of each type was sufficient for our results to stabilize.

Realistic examples. Rather than providing contrived counterexamples, every example we present comes from real data. To find these examples, we (algorithmically) searched the empirically-measured AS graph for specific subgraphs that could induce specific counterexamples, and then simulated the attack strategy.

Multiple datasets. Because the actual AS-level topology of the Internet remains unknown, and inferring AS relationships is still an active area of research, we run simulations on a number of different datasets: multiple years of CAIDA data [9], and Cyclops data [10] augmented with 21,000 peer-to-peer edges from [11]’s IXP dataset. Even though these datasets use different relationship-inference algorithms, the trends we observed across datasets were remarkably consistent. Thus, all the results and examples we present are from CAIDA’s November 20, 2009 dataset (with slight modifications to the sibling relationships, see Appendix A.2); counterparts of these graphs, computed from Cyclops and IXP data [10, 11] are in Appendix H.

3. FOOLING BGP SECURITY PROTOCOLS

This section overviews the security protocols we consider, and presents *the set of (possibly) bogus paths that a manipulator can announce to each without getting caught*. We use Figure 1 to demonstrate the fraction of traffic a manipulator could attract by announcing one of these (possibly) bogus paths to all its neighbors.

Our focus is on protocols with well-defined security guarantees. Thus, we consider the five major BGP security variants, ordered from weakest to strongest security, as follows: (*unmodified*) *BGP*, *Origin Authentication*, *soBGP*, *SBGP*, and *data-plane verification*. Because we focus on security guarantees and not protocol implementation, we use these as an umbrella for many other proposals (see [4] for a survey) that provide similar guarantees using alternate, often lower-cost, implementations. Furthermore, our ordering of protocols is strict: if a manipulator launches an attack that succeeds against a strong security protocol, then that same attack will also succeed against the weaker security protocol. We also consider an orthogonal mechanism, called *defensive filtering*.

BGP. BGP does not include mechanisms for validating information in routing announcements. Thus, the manipulator can get away with announcing any path he wants, including (falsely) claiming that he is the owner of the victim’s IP prefix. Indeed, when the manipulator AS 852 launches this attack on victim AS 3248’s IP prefix, our simulations show that he attracts traffic from 75% of the Internet.³

Origin Authentication. Origin authentication [5] uses a trusted database to guarantee that an AS cannot falsely claim to be the rightful owner for an IP prefix. However, the manipulator can still get away with announcing any path that *ends* at the AS that rightfully owns the victim IP prefix. For instance, in Figure 1, the manipulator AS 852 can attract traffic from 25% of the Internet by announcing the path (852, 3248, Prefix), even though no such path physically exists.

soBGP. Secure Origin BGP (soBGP) [6] provides origin authentication as well as a trusted database that guarantees that any announced path *physically exists* in the AS-level topology of the Internet. However, a manipulator can still get away with announcing a path that *exists* but is not actually *available*. In Figure 1, the manipulator AS 852 can attract traffic from 10% of the Internet by announcing the path (852, 3303, 3248, Prefix). Notice that this path is unavailable; **GR2** forbids AS 3303 to announce a peer path to another peer.

Of course, finding paths that exist in the AS graph requires the manipulator to have knowledge of the global topology of the network. However, obtaining this information is not especially difficult; an industrious manipulator might even obtain this information from the AS graph datasets [9, 10, 11] that we used in this paper, or

³In fact, another strategy, called a *subprefix hijack*, is available to manipulator; by announcing a longer, more specific subprefix of the victim’s IP prefix, he can attract traffic from 100% of the Internet. This work does not consider subprefix hijacks, both because these attacks are well understood, and because they can be prevented by the filtering practices discussed in [4].

even (ironically) from the soBGP database itself!

S-BGP. In addition to origin authentication, Secure BGP [7] also uses cryptographically-signed routing announcements to provide a property called *path verification*. Path verification guarantees that every AS a can only announce a path abP to its neighbors if it has a neighbor b that *announced* the path bP to a . Thus, it effectively limits a single manipulator to announcing available paths. For instance, in Figure 1, the manipulator’s *normal path* (see Section 2.3) is the five-hop customer path (852, 577, 2914, 286, 3248, Prefix); announcing that path allows him to attract traffic from 0.9% of the ASes in the Internet. However, with S-BGP the manipulator could instead announce the *shorter* four-hop provider path (852, 1239, 286, 3248, Prefix), thus doubling attracted traffic to 1.7%. Indeed, S-BGP does *not* prevent the manipulator from *announcing* the shorter, more expensive, provider path, while actually *forwarding* on the cheaper, longer customer path.

Data-plane verification. Data-plane verification [8, 4] prevents an AS from announcing one path, while forwarding on another. Thus, if the manipulator in Figure 1 wants to maximize his attracted traffic, he must also forward traffic on the provider path.

Defensive Filtering. Defensive filtering polices the BGP announcements made by stub ASes. Thus, defensive filtering requires *each provider* to keep track of the IP prefixes owned by its stub customers. If a stub announces a path to *any* IP prefix that it does not own, the provider drops/ignores the announcement. Defensive filtering *completely eliminates* attacks by stubs; in our model, since a stub has no customers, by **GR2** it should never announce paths to prefixes that it does not own. However, it does not prevent attacks by non-stubs.

Anomaly Detection. Anomaly detection mechanisms are outside our scope. Firstly, many of these provide functionalities that approximate the security guarantees described above, so their effectiveness is upper-bounded by the schemes we evaluate. Secondly, the remaining protocols usually do not have well-defined guarantees; *e.g.*, [15, 17] flag suspicious routes as potential export policy violations, but do not *guarantee* the detection of *every* export policy violation.

4. SMART ATTRACTION ATTACKS

We simulate attraction attacks on measured graphs of the Internet’s AS-level topology [9, 10, 11] to determine how much traffic a manipulator can attract in the *average case*. This section first presents the attack strategies we simulated, and then reports our results.

4.1 A smart-but-suboptimal attack strategy.

We assumed that ASes make routing decisions based

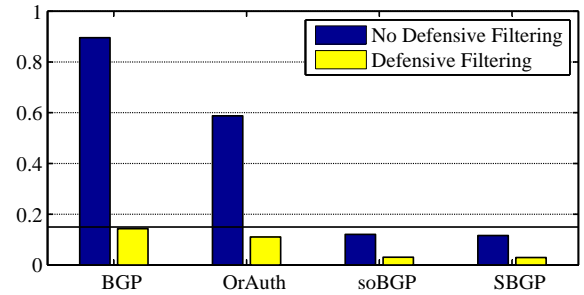


Figure 2: Lower bounds on the probability of attracting at least 10% of ASes in the Internet.

on business relationships and path length, and that a manipulator m cannot lie to his neighbor a about their business relationship (*i.e.*, between m and a). Thus, intuition suggests that the manipulator’s best strategy is to widely announce the shortest possible path:

“Shortest-Path Export-All” attack strategy. Announce to *every* neighbor, the *shortest* possible path that is *not flagged as bogus by the secure routing protocol*.

Every “Shortest-Path Export-All” attack strategy on S-BGP is also an attack on data-plane verification. The “Shortest-Path Export-All” attack strategy on S-BGP has the manipulator announce his shortest *legitimate available path* to the victim, instead of his *normal path* (see Sections 2.3 and 3). Notice that if the manipulator actually decides to *forward* his traffic over the announced path, he has a successful attack data-plane verification as well! Thus, the “Shortest-Path Export-All” attack strategy on data-plane verification is *identical* to the attack on S-BGP. (To reduce clutter, the following mostly refers to the attack on S-BGP.)

We underestimate damage. Section 6 shows that the “Shortest-Path Export-All” attack strategy is *not* actually optimal for the manipulator, and in Section 7 shows that finding the optimal attack strategy is NP-hard. Thus, we give up on finding the *optimal* attack strategy, and run simulations assuming that the manipulator uses this smart-but-suboptimal attack. This means that the results reported in this section *underestimate* the amount of damage a manipulator could cause, and we usually *cannot* use these results to directly compare different secure routing protocols. In spite of this, our simulations do provide both (a) useful lower bounds on the amount of damage a manipulator could cause, and (b) a number of surprising insights on the strategies a manipulator can use to attract traffic to his network.

4.2 Defensive filtering is crucial.

Our first observation is that defensive filtering is a crucial part of any Internet security solution:

Figure 2: We show the probability that, for a randomly chosen (manipulator, victim) pair, the manipulator can attract traffic destined for the victim from at least 10% of the ASes in the Internet. The manipulator uses the “Shortest-Path Export-All” attack strategy. The first four bars on the left assume that network does *not* use defensive filtering. We show the success of the manipulator’s strategy on each of the four BGP security variants, in network with and without defensive filtering of attacks by stubs. The horizontal line in Figure 2 shows the fraction of attacks that are completely eliminated by defensive filtering; since 85% of ASes in the CAIDA graph are stubs, properly-implemented defensive filtering guarantees that only 15% of manipulators can successfully attack any given victim.

Despite the fact that our experiments used *sub-optimal* strategies for the manipulator, Figure 2 leads to two concrete observations:

1. Even if we assume the manipulator runs the *sub-optimal* “Shortest-Path Export-All” attack strategy on a network that has S-BGP (or data-plane verification) but not defensive filtering, he can still attract 10% of the Internet with probability 10%; furthermore, it may be possible that more clever strategies for S-BGP (*e.g.*, Figure 11-Figure 12) can increase the manipulator’s probability of success to the point where simple defensive filtering, which is guaranteed to eliminate all but 15% of attacks, performs even better than S-BGP.
2. Even if both S-BGP (or data-plane verification) *and* defensive filtering are used, there is still a non-trivial 2% probability that the manipulator can attract 10% of the Internet. Better attack strategies could increase this probability even further. This is particularly striking when we compare with the normal case, where the manipulator manages to attract 10% of the Internet with about 10^{-4} probability.

4.3 Attack strategy on different protocols.

The reader may wonder why we chose to focus specifically on the probability of attracting 10% of the Internet in Figure 2. In the interest of full disclosure, we now present the full picture:

Figure 3: We show the complimentary cumulative distribution function (CCDF) of the probability that at least a x -fraction of the ASes in the Internet forward traffic to the manipulator when he uses the “Shortest-Path Export-All” attack strategy. Probability is taken over the uniform random choice of a victim and manipulator, and observe that Figure 2 simply presents a cross-section of these results at the x-axis value of $x = 10\%$. Because this figure carries quite a lot of information, we walk through a few interesting points:

BGP curve. Here, the manipulator *originates*, *i.e.*, announces that he is directly connected to, the victim

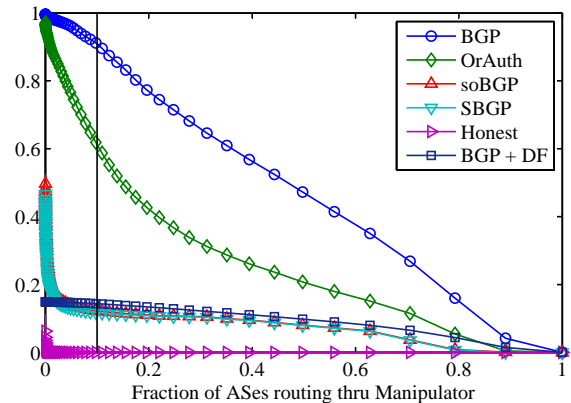


Figure 3: CCDF for the “Shortest-Path Export-All” attack strategy.

prefix. This curve looks almost like the CCDF of a uniform distribution; this makes sense, because the manipulator, and the AS that legitimately owns the victim prefix, both announce one-hop paths to the prefix. Thus, they are on equal footing in a ‘tug-of-war’ to attract traffic from the rest of the Internet.

Origin Authentication curve. This time the manipulator announces that he has a direct link to the AS that legitimately owns the victim prefix. Because the manipulator’s path is now two hops long, the amount of traffic he can attract on average is reduced.

soBGP and S-BGP/data-plane verification curves.

For the attack on soBGP, the manipulator announces the shortest path that *exists* in the AS graph. For the attack on S-BGP (and data-plane verification), the manipulator announces the shortest *available* path that he learned from his neighbors. The soBGP and S-BGP curves are almost identical, which seems strange, since S-BGP provides stronger security guarantees than soBGP. We discuss this further in Section 4.4. For now, however, notice that both curves drop off sharply, with a knee around $x = 2\%$, $y = 15\%$. This means that 85% of manipulations do not manage to attract more than 2% of the Internet; these numbers roughly correspond to the fact that 85% of ASes in the graph are stubs that fail to attract much traffic with the “Shortest-Path Export-All” attack strategy on soBGP and S-BGP. Meanwhile, between $x = 2\%$ to $x = 60\%$, both curves tend to flatten out, suggesting that if a manipulator is able to attract at least 2% of the Internet, he is almost equally likely to be able to attract 60%. We spend more time on this observation in Section 4.5.

Honest curve. Here the manipulator behaves ‘normally’; that is, using the ranking and export policies described in Section 2.2. Observe that this curve looks almost like a delta-function at $x = 0$. That is, a randomly-chosen AS is likely to attract only a negligible fraction of the Internet by behaving normally.

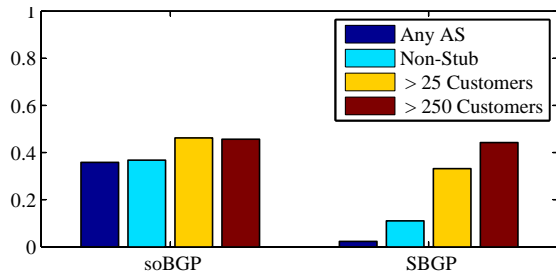


Figure 4: Probability of finding a shorter path.

BGP+Defensive Filtering curve. Here, defensive filtering is used to eliminate all the “Shortest-Path Export-All” attack strategies on BGP launched by stubs, *i.e.*, by 85% of ASes. Thus, the ‘BGP+DF’ curve is simply the ‘BGP’ curve scaled down to about 15%.

4.4 S-BGP forces long path announcements.

Figure 2 - 3 shows that S-BGP is *not* much more effective in preventing “Shortest-Path Export-All” attack strategies than the less-secure soBGP. To understand why, let’s compare the lengths of the path that the manipulator can announce with soBGP and S-BGP:

Figure 4: We show the probability that the manipulator can announce a path that is shorter than the *normal path*, *i.e.*, that path he would have chosen if had used the rankings in Section 2.2. Probability is taken over a randomly-chosen victim, and a manipulator that is randomly chosen from one of the following four *classes*: (a) Any AS in the graph, (b) *Non-stubs*, or ASes with at least one customer (c) Medium-sized ASes with at least 25 customers, and (c) Large ASes with at least 250 customers. If we focus on the results for S-BGP, it is clear that *larger* ASes are more likely to find *shorter* paths through the network; this follows from the fact that these ASes are both more richly connected (*i.e.*, they have large degree), as well more central (*i.e.*, they are closer to most destinations in the Internet). Furthermore, we can also see that ASes (especially small ASes) are more likely to find short paths with soBGP than they are with S-BGP.

From Figure 4, we can conclude that S-BGP is doing exactly what it is designed to do: it is limiting the set of paths from which the attacker can announce, forcing him to announce longer paths. However, in light of the results in Figures 2-3, we must ask ourselves why forcing the manipulator to announce longer paths does not seem to significantly limit the amount of traffic he attracts. We could explain by arguing that path lengths in the Internet are fairly short, (averaging about 5 hops in our simulations, see Appendix C); else (averaging about 5 hops in our simulations); so the paths that the manipulator can get away with announcing in soBGP are only a few hops shorter than the paths he can announce with S-BGP. While this is true, the next section

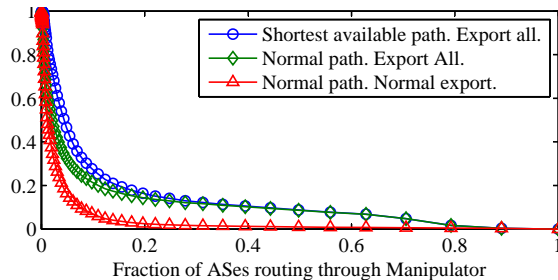


Figure 5: Aggressive export policies.

shows that there is an even deeper issue here; namely, that the *length* of the manipulator’s path often plays less of a role than the *set of neighbors that he exports to*.

4.5 Length matters less than export policy.

We now show that the attacker’s export policy can play a more important role than the length of the path he announces:

Figure 5: We show another CCDF of the probability that at least a x -fraction of the ASes in the Internet forward traffic to the manipulator; probability is taken over a randomly-chosen victim, and a manipulator chosen randomly from the class of ASes that have at least 25 customers. We consider three different strategies: (a) Announce the shortest available path to all neighbors (equivalent to the “Shortest-Path Export-All” attack strategy on S-BGP) (b) Announce the normal path to all neighbors (c) Announce the normal path using the normal (**GR2** and **NE**) export policy.

This figure clearly shows that, on average, announcing a *shorter* path is much less important than announcing a path to more neighbors (*i.e.*, the curves for (a) and (b) are very close, while the curves for (b) and (c) are quite far apart). Indeed, when we considered at smaller manipulators (not shown), the curves for (a) and (b) are even closer together. To explain this, consider the following: by violating the normal export policy, the manipulator can announce paths to his providers, even when he does not forward traffic over a customer path. The providers are more likely to choose the customer path through the manipulator, over some possibly shorter, non-customer path that they might have, and so the number of customer paths through the network that are incident on the manipulator increases. Since every AS prefers a customer path over a non-customer path, customer paths tend to carry a large volume of traffic to the manipulator.

Thus, Figure 5 teaches us that it is often more important for the manipulator to create *customer* paths than to create *short* paths.

4.6 Tier 2s usually cause the most damage.

Before we conclude this section, we would like to de-

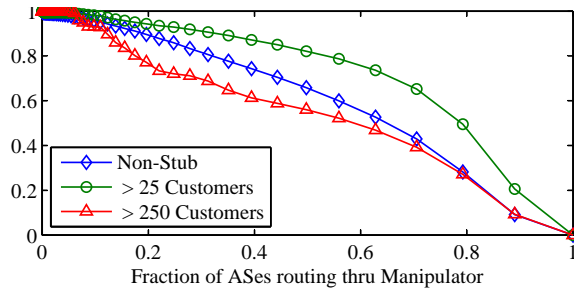


Figure 6: “Shortest-Path Export-All” attack strategy on BGP by different manipulators.

termine which ASes in the Internet are likely to be the most successful manipulators. We consider non-stub manipulators from three different classes: (a) ASes with at least one customer (b) ASes with at least 25 customers, (roughly modeling “Tier 2 ASes”) and (c) Large ASes with at least 250 customers (“Tier 1 ASes”):

Figure 6: We once again show a CCDF of the probability that at least a x -fraction of the ASes in the Internet forward traffic to the manipulator, when the manipulator launches the “Shortest-Path Export-All” attack strategy on BGP. Despite the fact that the “Tier 1” manipulators are more central than the “Tier 2s”, we make the surprising observation that “Tier 2s” manage to attract more traffic than than “Tier 1s”. In fact, for certain regimes, even smaller non-stub ASes tend to attract more traffic than the “Tier 1s”!

This bizarre observation is actually easy to explain. In the “Shortest-Path Export-All” attack strategy on BGP, every manipulator (regardless of its size or location in the network) announce a single-hop path to the victim prefix. Thus, announced path length does *not* play a role when we compare across classes of manipulators. On the other hand, despite their centrality, Tier 1 ASes are more expensive to route through than every other AS in the Internet; a Tier 1 is always a provider or peer of its neighbors, so even if those neighbors learn a short path through the Tier 1, they will prefer to route over a (potentially longer) path through one of their own customers. Furthermore, Tier 2’s more central and richly connected than smaller ASes on the edge of the Internet, and thus they tend to attract more on average than the smaller non-stub.

The reader may be troubled by the fact that the (red triangle) curve for the manipulators with at least 250 customers has a different shape than the other curves in Figure 6. We saw exactly this effect on all our experiments across different datasets, and one main reason it occurs is because the AS graph we used only has 34 ASes (out of a total of 33K ASes) that have at least 250 customers; this is consistent with the idea that are about 12 (or so) Tier 1 ASes in the Internet. Because we had so few manipulators to choose from, the effect

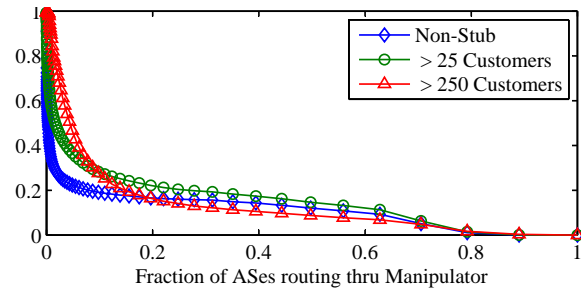


Figure 7: “Shortest-Path Export-All” attack strategy on S-BGP/data-plane verification by different manipulators.

of individual manipulators on the results become more pronounced, and the curves become less ‘smooth’.

4.7 S-BGP is vulnerable to non-stubs.

The picture for origin authentication looks about the same Figure 6. However, the results change when we look at soBGP and S-BGP/data-plane verification. Since, as usual, our results for soBGP and S-BGP look about the same, we only discuss S-BGP:

Figure 7: This is the CCDF for S-BGP/data-plane verification (*cf.*, to Figure 6). “Tier 2” manipulators usually come out on top, except when we consider manipulations that attract 10% of the Internet or less. In this regime, the Tier 1 ASes come out on top, so that the S-BGP curve mimics normal behavior (not shown). Tier 1s tend to attract more traffic than others when everyone is normal, because they are likely to have short customer paths they can announce to all of their (many) neighbors.

4.8 Summary.

In some sense, the results of this section suggest that secure routing protocols like S-BGP and soBGP are only dealing with one half of the problem: while they do restrict *the path* the manipulator can choose to announce, they fail to restrict his *export policies*. Indeed, this observation has a number of implications:

1. Because defensive filtering restricts both the export policies and the paths announced by stubs, we find that it provides a level of protection that is at least comparable to that provided by S-BGP, and even data-plane verification, alone.
2. Even if we eliminate attacks by stubs, Figures 6 - 7 show that the Internet is still vulnerable to non-stub ASes that both (a) deviate from normal routing policies by announcing shorter paths, and (b) deviate from normal export policies by announcing non-customer paths to *all* their neighbors. Furthermore, more clever export policies (rather than simply exporting to all) to could lead to better attacks, (*e.g.*, Figure 12).
3. By deviating from normal export policies, Tier 2s

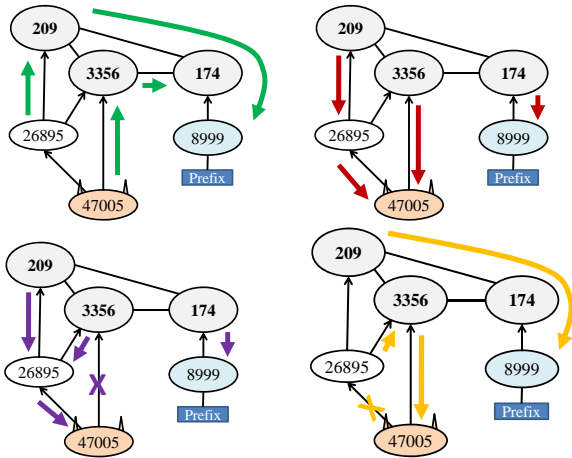


Figure 8: (a) Normal outcome. (b)-(d) Black-hole.

can launch the most effective “Shortest-Path Export-All” attack strategies.

5. SMART INTERCEPTION ATTACKS

We now turn our attention to traffic interception attacks [1, 2, 4]. In an interception attack, the manipulator would like to attract as much traffic as possible to his network (in order to eavesdrop or tamper with traffic) before forwarding it on to the victim IP prefix. Thus, we require that an interception attack *preserves an available path from the manipulator to victim*.

5.1 A stub that creates a blackhole.

To provide some intuition, we first show how a manipulator could lose a working path to a victim:

Figure 8: For simplicity, let’s consider an attack on BGP where the manipulator falsely originates the victim’s prefix. The manipulator, AS 47005, a web-hosting company in Illinois, wants to attract traffic destined for the victim, AS 8999, a web-hosting company in France. The manipulator is a multi-homed stub with two providers, Level 3 Communication’s AS 3356, and AS 26895, a Chicago-area telecom provider. The left figure shows the normal outcome, where the manipulator has a path to victim available through each of his providers. The right figure shows what happens when the manipulator announces the victim’s prefix to each of his providers; since each of them prefer short customer paths, they will forward their traffic through the manipulator. The manipulator has now created a *black-hole*; he has no available path to the victim AS 8999 through either of his providers. Suppose now that the manipulator tried to be a little more clever, and did *not* announce the victim’s prefix to his Tier 1 provider AS 3356. Unfortunately for the manipulator, this strategy still creates a blackhole. As show in the bottom left (purple) figure, AS 3356 will prefer his customer path

through manipulator (3356, 26895, 47009, Prefix) over his peer path to the legitimate prefix (3356, 174, 8999, Prefix). Thus, both the manipulator’s providers will still forward their traffic to the manipulator, and the blackhole remains. It is easy to see that a blackhole also occurs when the manipulator only announces the victim prefix to his local provider AS 26895 (see the bottom right (orange) figure).

5.2 When do interception attacks succeed?

The reader may be surprised to learn that are many situations in which blackholes are *guaranteed not to occur*. We can prove that, within our model of routing policies, the manipulator can aggressively announce paths to certain neighbors while still preserving a path to the victim:

THEOREM 5.1. *Assume that **GR1** holds, and that all ASes use the routing policies in Section 2.2. Suppose the manipulator has an available path through a neighbor of a type x in the normal outcome. If there is \checkmark in entry (x, y) of Table 1, then a path through that neighbor will still be available, even if the manipulator announces any path to any neighbor of type y .*

Appendix G presents the proofs, and we note that the results marked with \checkmark^* do *not* require **GR1** to hold. This theorem is ‘sharp’, in the sense that if there is an X in entry (x, y) of Table 1, we can show that manipulator *can sometimes* lose an available path of type x if he announces certain paths to a neighbor of type y ; indeed, Figure 8 is proof of the X in the lower-right entry of Table 1. While results of this form were presented in an earlier work [1], their work claims that a peer-path *cannot* be lost by announcing to a provider (and vice versa). Appendix E presents an example contradicting [1]’s claim, and proving the remaining X entries in Table 1.

Tier 1s and Stubs. Theorem 5.1 leads to a number of observations, also noted by [1]. First, interception is easy for Tier 1s. Since Tier 1s have no providers, they need only concern themselves with the four upper-left entries in Table 1, which indicate that they can announce paths to all their neighbors. Secondly, interception is hard for stubs. A stub’s neighbor is always a provider, putting it in the bottom-right entry of Table 1, indicating that aggressive announcements could cause a blackhole (*e.g.*, Figure 8).

5.3 When do “Shortest-Path Export-All” attack strategies cause a blackhole?

The observations of Section 5.2-5.4 are borne out by our experiments. Recall that in the “Shortest-Path Export-All” attack strategy on a BGP security variant, the manipulator announces his shortest (non-rejected) to *all* of his neighbors. We now show, that this simple

To preserve a path of type...	May announce to neighboring...		
	Customers	Peers	Providers
Customer	✓*	✓*	✓
Peer	✓*	✓*	X
Provider	✓	X	X

Table 1: Guidelines for interception.

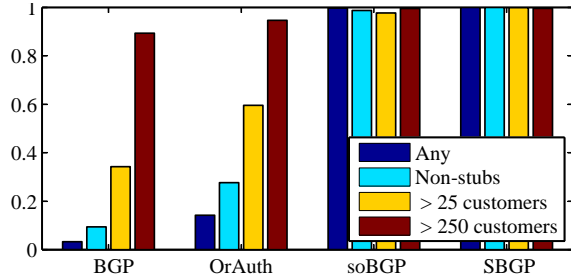


Figure 9: Probability that the “Shortest-Path Export-All” attack strategy does *not* create a blackhole.

attack strategy often allows the manipulator to intercept traffic without creating a blackhole:

Figure 9: We show the probability that the manipulator has some available path to the victim if he uses the “Shortest-Path Export-All” attack strategy for each of the four BGP security variants. We present results for a randomly-chosen victim, and a manipulator chosen from the usual four classes (see Figure 4). We assume that manipulator runs the “Shortest-Path Export-All” attack strategy on each BGP security variant. We can make a number of observations:

1. Manipulators with the *most* customers are *least* likely to create a blackhole. As discussed in Section 5.2, these manipulators are most likely to have an available customer path to the victim, and as shown in the first row of Table 1, can get away with announcing to *all* their neighbors without creating a blackhole.
2. The attack on BGP is most likely to cause a blackhole (*cf.*, the attack on origin authentication, or soBGP). Because the manipulator announces a more attractive (*i.e.*, short) path in his attack on BGP, he is more likely to convince *all* of his neighbors to forward traffic to him, and thus create a blackhole.

We note that our empirical results generally agree with Theorems 5.1; whenever there was a gap between the two, we found a customer-provider loop (*i.e.*, a violation of **GR1**) in the AS graph that we used for running our simulations. We are not particularly troubled by this gap, since the algorithms used to produce AS relationship graphs from empirical data [9, 10] sometimes introduce artifacts like customer-provider loops.

5.4 Interception by announcing available paths.

Figure 9, and other simulation results (not shown)

also indicates that the “Shortest-Path Export-All” attack strategy on S-BGP, *never* creates a blackhole (as long as the manipulator had a path to the victim in the normal outcome). This observation matches intuition; since S-BGP forces the manipulator to announce an available path, the manipulator must of course have an available path to the victim! Indeed, we conjecture that it is possible to prove a more general statement that implies *every* successful attraction attack strategy on S-BGP is also an interception attack. That is, suppose ASes use the routing policies in Section 2.2 and **GR1** holds, and consider *any* path P that is available to the manipulator in the normal outcome. Then path P remains available if the manipulator announces P to *any* subset of his neighbors. We leave the proof of such a statement to future work.

5.5 Two interception strategies.

Figure 9 immediately suggests a simple interception strategy that seems to work every time:

“Shortest-Available-Path Export-All” attack strategy: The manipulator should announce his shortest *available* path from the normal outcome to all his neighbors. Recall that this is exactly the “Shortest-Path Export-All” attack strategy on S-BGP.

Figure 3, shown that this strategy attracts more traffic than the normal strategy, but also suggests that when the network does *not* use S-BGP, there may be better interception attack strategies. Indeed, Figure 9 shows that there is a non-trivial probability that the manipulator has an available path to the victim, even if he launches the “Shortest-Path Export-All” attack strategy on the BGP. This suggests the following simple strategy:

“Hybrid Interception” attack strategy: First, run the “Shortest-Path Export-All” attack strategy on the secure routing protocol, and check if there is an available path to the victim. If no such path is available, announce the shortest path that was available in the normal outcome to all neighbors.

By no means do we believe that these two strategies are optimal; indeed, we evaluated more clever attack strategies, but decided to omit them from this paper in the interest of brevity and simplicity. What is surprising is that even these trivial strategies can be quite effective for certain manipulators.

5.6 Evaluating interception strategies.

From the discussion above (Figures 8- 9, Section 5.2), it is clear that ASes with very few customers are unlikely to attract large volumes of traffic without blackholing themselves. For this reason, we focus our evaluation on manipulators with at least 25 customers, and for brevity only present attacks on BGP:

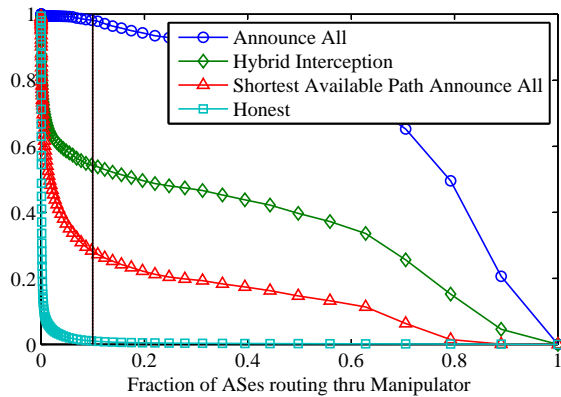


Figure 10: Interception attacks on BGP.

Figure 10: This is a CCDF of the probability that at least a x -fraction of the ASes in the Internet forward traffic to the manipulator, under the assumption that the network uses BGP. We compare the (a) “Shortest-Path Export-All” attack strategy where the manipulator *is allowed to create a blackhole*, with (b) the two interception strategies above, as well as (c) the normal strategy. Our key observation here is that the “Hybrid Interception” attack strategy manages to intercept a large fraction of traffic; *e.g.*, at least 10% of the Internet with probability over 50%!

5.7 Summary.

On average, traffic interception is difficult for stubs, but a manipulator with many customers can quite easily launch an interception attack. Indeed, manipulators with many customers can intercept a large volume of traffic with even the highly non-optimal “Hybrid Interception” attack strategy. Furthermore, as we shall discuss in Section 6, there may be more clever traffic interception attacks that allow the manipulator to attract even larger portions of the Internet, and some of these strategies may even work for stubs (*e.g.*, Figure 11)!

6. SMART ATTACKS ARE NOT OPTIMAL

We now prove that the “Shortest-Path Export-All” attack strategy is *not* optimal for the manipulator. We present three surprising counterexamples⁴, found in CAIDA’s AS graph, that show that (a) announcing *longer* paths can be better than announcing shorter ones, (b) announcing to *less* neighbors can be better than to announcing to more, and (c) the *identity* of the ASes on the announced path matters, since it can be used to *strategically trigger BGP loop detection*. In fact, (c) also proves that announcing a longer path can be better than a prefix hijack (where the manipulator originates a prefix he does not own)!

⁴Notice how each example was chosen to contradict the optimality of one aspect of the “Shortest-Path Export-All” attack strategy.

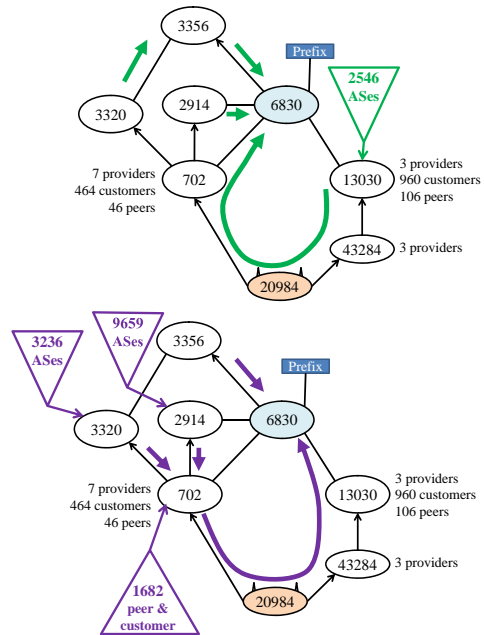


Figure 11: Announcing a longer path.

6.1 Attract more by announcing longer paths!

Our first example is for a network with soBGP, S-BGP or data-plane verification. We show a manipulator that *triples* his attracted traffic by announcing a *legitimate path to the victim, that is not his shortest path*. In fact, this strategy is so effective, that it attracts almost as much traffic as an aggressive prefix hijack on unmodified BGP!

Figure 11: The manipulator, AS 20984, is a small stub AS in Basel, Switzerland, that has one large provider, AS 702 owned by Verizon and having degree over 500, and one small provider, AS 43284 owned by Industrielle Werke Basel and having degree only four. The victim is European broadband provider AS 6830.

Prefix hijack. In a network with (unmodified) BGP, the manipulator could run a simple prefix hijack, announcing “20984, Prefix” to both his providers, and attract traffic from 62% of the Internet, exactly 20550 ASes. However, this strategy both creates a blackhole at the manipulator, and fails against soBGP or S-BGP.

Naive strategy. The upper (green) figure shows the “Shortest-Path Export-All” attack strategy, where the manipulator naively announces a *three-hop* available path, (20984, 702, 6830, Prefix) to his provider 43284. Since ASes 43284 and 13030 prefer the customer path to that manipulator, over their existing peer paths, both will forward traffic to the manipulator. He intercepts traffic from 16% of the Internet, or exactly 5569 ASes.

Clever strategy. The lower (purple) figure shows the manipulator cleverly announcing a *four-hop* available path (20984, 43284, 13030, 6830, Prefix) to his provider AS 702. AS 702 will prefer the longer cus-

tomers path through the manipulator over his shorter peer connection to AS 6830, but this time, the manipulator *triples* the amount of traffic he attracts, intercepting traffic from a total of 56% of the Internet, or exactly 18664 ASes. In fact, by announcing a longer path, the manipulator earns almost as much traffic as the aggressive prefix hijack.

Why it works. Notice that the manipulator’s provider Verizon’s AS 702 has hundreds more neighbors than his other provider, AS 43284, and that the clever strategy attracts Verizon’s AS 702 while the naive strategy attracts AS 43284. Attracting traffic from the larger AS is crucial to the manipulator’s success; in fact, it is more important than announcing short paths.

Details. Figure 11 shows that in the naive (green) strategy, Verizon’s two providers AS 3320 and AS 2914 route along *peer* paths that do not go through the manipulator, and can thus announce paths to their customers only. On the other hand, in the clever strategy, Verizon’s two providers, use *customer* paths through the manipulator; as such, they can announce paths to their customers, peers and providers, and each carry a large volume of traffic (from almost 13K ASes). Thus, in clever (purple) strategy, the manipulator attracts traffic from almost 1.7K ASes that route through Verizon’s AS 702 along customer or peer paths, as well as 13K ASes that route through Verizon’s providers, ASes 3320 and 2914. On the other hand, in the naive (green) strategy, the manipulator attracts traffic from about 2.5K ASes that are AS 13030’s customers, peers and providers; these 2.5K ASes do not route through the manipulator when he uses the clever (purple) strategy. Thus, in the naive (green) strategy, the attacker gains traffic from 2.5K ASes, while in the clever (purple) strategy on the right, the manipulator gains traffic from about $3.2K + 9.6K + 1.7K$ ASes, for a difference of $3.2K + 9.6K + 1.7K - 2.5K = 12K$ ASes. The basically accounts for the fact that the naive strategy on the right attracts traffic from only 5K nodes, while the clever strategy attracts traffic from 18K nodes.

When it works. This strategy only involves deviating from normal export policy, rather than *lying* about paths. Thus, it succeeds against *any* secure routing protocol (except when it is launched by stubs in a network with defensive filtering).

6.2 Attract more by exporting less!

This example is for a network with origin authentication, soBGP, S-BGP, data-plane verification, and/or defensive filtering. We show a manipulator that intercepts 25% *more* traffic by exporting to *less* neighbors.

Figure 12: The victim is AS 29993, a stub serving a liberal arts college in Illinois. The manipulator is AT&T’s AS 7132, and is in competition with the victim’s other provider AS 6325, a local ISP in Illinois.

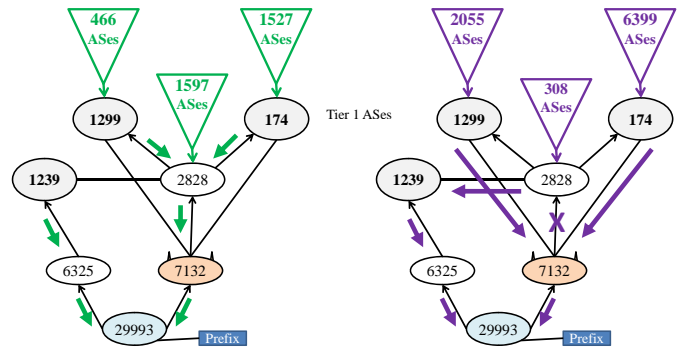


Figure 12: Exporting less.

The manipulator wants all the traffic destined for the victim to flow through his own AS.

Naive strategy. The “Shortest-Path Export-All” attack strategy requires the manipulator to announce his path to all his neighbors. On the left, when the manipulator announces a path to provider AS 2828, both AS 2828 and its two Tier 1 providers will route through the manipulator. As a result, the two Tier 1’s use *four-hop* paths to the victim, and the manipulator attracts traffic from 40% of the Internet, *i.e.*, 13463 ASes.

Clever strategy. On the right, the manipulator increases his traffic volume by almost 25%, by *not* exporting to his provider AS 2828. Because AS 2828 no longer has a customer path to the victim, he is forced to use a peer path through AS 1239. Because AS 2828 now uses a peer path, he will not export a path to to the two Tier 1 ASes 1299 and 174. The Tier 1s are now forced to choose a shorter *three-hop* peer path to victim through the manipulator. Because the two Tier 1’s now announce shorter paths to their customers, they become more attractive to the rest of the Internet, the volume of traffic they send to the manipulator quadruples, and the manipulator attracts 50% of the Internet, *i.e.*, 16658 ASes.

Why it works. The manipulator’s strategy forces influential ASes (*i.e.*, Tier 1s) to choose *shorter* peer paths over *longer* customer paths. He does this by *suppressing* announcements to certain providers, thus eliminating certain customer paths from the Internet.

Details. To account for change in traffic through the manipulator, notice that in the naive (green) strategy, AS 7132 attracts traffic from about 0.5K ASes that route through Tier 1 AS 1299 and 1.5K ASes that route through Tier 1 AS 174, and 1.6K other ASes that route through AS 2828. In the clever (purple) strategy on the right, the two Tier 1’s announce shorter paths and attract traffic from a total 8.4K ASes. Meanwhile, AS 2828, who no longer forwards traffic through the manipulator, only attracts traffic from about 300 ASes; this sharp decrease in traffic flowing through AS 2828 follows from the fact that in the clever strategy, AS2828 uses a *peer* path to the victim, and thus will no longer

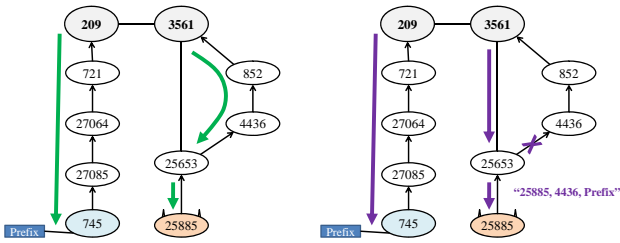


Figure 13: False loop prefix hijack.

accept traffic from its peers and providers. Thus, in the clever strategy, the attacker gains traffic from 8.4K - .3K ASes, and in the naive strategy the attacker gains traffic from .5K + 1.5K + 1.6K ASes, for a difference of 4.5K ASes; this roughly accounts for the 25% increase in traffic that the manipulator attracts by using the clever strategy.

When it works. This strategy only involves using a clever export policy, rather than lying about paths, and therefore succeeds against *any* protocol, including data-plane verification.

6.3 Attract more by gaming loop detection!

To show that the identity of the ASes on the announced path can affect the amount of attracted traffic, our last example involves gaming BGP loop detection. While gaming loop detection was explored in other works, *e.g.*, [2, 4, 3], what is remarkable about this example is that it proves that this attack strategy *can attract more traffic than an aggressive prefix hijack*.

Figure 13: The manipulator is AS 25885, a stub in Clifton, NJ with two providers. This figure only depicts his NJ-area provider, Fortress ITX AS 25653. The manipulator wants to blackhole traffic destined for AS 745, a stub in Alabama.

Standard prefix hijack. The manipulator announces the path (25885, Prefix) and attracts traffic from the most of the Internet, exactly 32010 ASes. Notice also that because AS 3561 prefers customer paths, this large AS will chose to forward his traffic along the *five-hop* customer path through the manipulator.

False loop prefix hijack. The manipulator claims that *innocent* AS 4436 originates the prefix, announcing (25885, 4436, Prefix) to Fortress ITX AS 25653. However, when this *false loop* is announced to AS 4436, BGP loop detection will cause AS 4436 to reject the path through Fortress ITX AS 25653. As a result, AS 3561 has no customer path to the prefix, and instead chooses to forward traffic along the shorter peer path. Now, AS 3561 announces a shorter, *four-hop* path to his neighbors (3561, 25653, 25885, 4436, Prefix), making him more attractive to the rest of the Internet, and attracting more traffic to the manipulator. For this, and other reasons that are discussed in Appendix D, the manipulator attracts 360 more ASes than standard

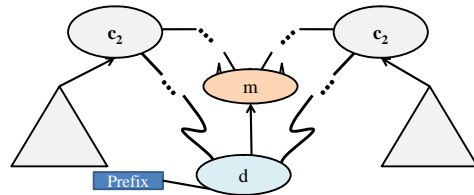


Figure 14: DILEMMA for proving hardness.

prefix hijack, *i.e.*, 32370 ASes.

Why it works. The manipulator games BGP loop detection, effectively ‘removing edges’ from the network, to force influential ASes (*i.e.*, Tier 1s) to chose shorter peer paths over longer customer paths.

When it works. This strategy involves lying about the path an innocent AS announces to the victim, in order to trigger BGP loop detection. Because S-BGP and data-plane verification prevent lying about the paths, this strategy only works with BGP, origin authentication, or soBGP.

7. FINDING OPTIMAL ATTACKS IS HARD

After all the bizarre attack strategies in Section 6, the reader might not be surprised by the following:

THEOREM 7.1. *If ASes use the routing policies of Section 2.2, then finding a manipulator’s optimal traffic attraction attack strategy is NP-hard.*

This theorem holds for (a) any of secure protocols variants and (b) also extends to interception attacks; our proof, which is sadly relegated to Appendix F, uses a reduction to the problem of finding the maximum independent set in a graph.

In Appendix F, we also show that it is hard to even *approximate* the optimal attack within a constant factor; thus we cannot even design an algorithm that gets “close” to the optimal attack on a general AS graph. These results imply that we are unlikely to be able to fully characterize the manipulator’s optimal attack strategy. Thus, any analysis of traffic attraction attacks will have to resort the techniques we used here — investigating smart but suboptimal attacks via specific observations about about the Internet AS graph.

Proof technique (Fig. 14). Our proof proceeds in two stages: we first use a special network gadget we call DILEMMA to reduce our problem to the problem of finding the maximum independent set in a graph. We then show how a DILEMMA can exist for different secure routing protocols and types of attacks we considered in this paper. In DILEMMA, the manipulator *m* wants to attract the traffic for the victim *d* from two influential ASes *c*₁ and *c*₂, whose carry traffic from the majority of the network. A DILEMMA construction must guarantee that *m* can attract each of the ASes individually, but cannot attract both ASes simultaneously. Details are in Appendix F.

Choosing a path to announce is hard. Our hardness results for attacks on BGP, origin authentication, and soBGP amount to showing that it is hard for the manipulator to decide which path he should announce to his neighbors. These results are meaningful even if the manipulator has only two (or more) neighbors (since the hardness is related to the number of paths through the network).

Choosing an export policy is hard. On the other hand, the reader might suspect that the finding the optimal attack strategy becomes *easier* if the manipulator is only allowed to announce an *available* path, as with S-BGP. Surprisingly, this is not the case; we show that even if the manipulator is forced to announce his *normal path*, it is still hard for him to choose *the optimal set of neighbors to announce paths to*. However, these hardness results are only meaningful when the node has a large number of neighbors.

8. RELATED WORK

Previous papers have proposed security extensions to BGP (see [4] for a survey). These papers typically use a particular attack model to analyze the proposed protocol, and compare it to BGP, but understandably do not address attacks outside of their model, like traffic-attraction attacks.

Recent theoretical work [3, 18] considers strategic attacks launched by economically-motivated ASes. These papers construct example topologies—sometimes quite contrived—where an AS can manipulate a particular variant of BGP. However, these papers do not define a specific attack strategy, investigate the optimality of attacks, or demonstrate whether the example topologies exist in practice. In contrast, we evaluate attacks on an empirically-measured AS-level topology, and show that our counterexamples are realistic by finding them in the AS level topology.

There have been many works that empirically investigate attacks on BGP (see [4] for a survey). Our work is most closely related to an earlier study of prefix-hijack and interception attacks [1]. While [1] focuses on (unmodified) BGP and two specific attacks (i.e., prefix-hijack and invalid-next-hop attacks), we consider attacks against a variety of secure routing protocols. We show that the attacks considered in [1] are suboptimal (Section 6.3), and prove that finding the the optimal attack is NP-hard. The work in [1] suggests guidelines for interception similar to the ones we present in Table 1. However, our guidelines correct an error in [1]’s earlier paper (see Section 5.2).

Our work is also related to earlier work [19], that compares several BGP security protocols under *partial* deployment. In contrast, we focus on a *full* deployment, using a model that captures realistic routing policies. However, [19] considers a simplified model that ignores

business relationships, and instead assumed that normal ASes prefer shortest paths and export paths to all neighbors. This simplification means that soBGP and S-BGP are the same within their model, making it difficult to compare across protocols.

9. CONCLUSIONS

Because we work within a *model* of routing policies, we caution against interpreting our results as *hard numbers* that measure the impact of an attack launched by a specific manipulator in practice. However, the *trends* uncovered by our quantitative analysis do allow us to arrive at a number of useful insights; indeed, many of these insights are obtained through *average case* analysis, and thus we suspect that they hold up even if some ASes deviate from the policies in our model. (Indeed, this study demonstrates that accurate models of the routing policies are important for assessing secure routing protocols.) Furthermore, the trends we identified were remarkably consistent across multiple AS topology datasets [9, 10, 11].

While secure routing protocols can blunt traffic attraction attacks, we found that *export policies* are very effective attack vector that these protocols do not cover. Thus, we suggest that secure routing protocols (*e.g.*, soBGP and S-BGP) should be deployed *in combination* with mechanisms that police export policies (*e.g.*, defensive filtering). We note, however, that policing export policies is a significant challenge in practice. Defensive filtering of stubs requires voluntarily compliance from each provider, and it is difficult to check for proper implementation (as evidenced by recent events [20]). Moreover, given the complexity of routing policies used in practice on the Internet, we lack even a *definition* of what it means to deviate from normal export policies. Thus, while anomaly-detection techniques that flag suspicious routes [15, 17] could help, tackling these issues remains an important avenue for future research.

Acknowledgments

The authors thank Jeff Lupien and Paul Oka for outstanding research assistance, and Boaz Barak, Randy Bush, Kevin Butler, Nick Feamster, Avinatan Hassidim, Elliott Karpilovsky, Dave Ward, Dan Wendlandt and the members of the MSR-New England Lab for useful discussions and comments.

10. REFERENCES

- [1] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the Internet," in *ACM SIGCOMM*, 2007.
- [2] A. Pilosov and T. Kapela, "Stealing the Internet: An Internet-scale man in the middle attack," Aug. 2008. Presentation at DefCon 16, <http://eng.5ninesdata.com/~tkapela/iphd-2.ppt>.
- [3] S. Goldberg, S. Halevi, A. D. Jaggard, V. Ramachandran, and R. N. Wright, "Rationality and traffic attraction: Incentives for honest path announcements in BGP," in *ACM SIGCOMM*, 2008.
- [4] K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," *Proceedings of the IEEE*, January 2010.
- [5] P. McDaniel, W. Aiello, K. Butler, and J. Ioannidis, "Origin authentication in interdomain routing," *Computer Networks*, Nov. 2006.
- [6] R. White, "Deployment considerations for secure origin BGP (soBGP)." draft-white-sobgp-bgp-deployment-01.txt, June 2003, expired.
- [7] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (S-BGP)," *J. Selected Areas in Communications*, vol. 18, pp. 582–592, April 2000.
- [8] E. L. Wong, P. Balasubramanian, L. Alvisi, M. G. Gouda, and V. Shmatikov, "Truth in advertising: Lightweight verification of route integrity," in *PODC*, 2007.
- [9] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, and kc claffy, "AS relationships: Inference and validation," *ACM SIGCOMM Computer Communication Review*, Jan. 2007.
- [10] Y.-J. Chi, R. Oliveira, and L. Zhang, "Cyclops: The Internet AS-level observatory," *ACM SIGCOMM Computer Communication Review*, Oct. 2008.
- [11] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: Mapped?," in *Proc. Internet Measurement Conference*, Nov. 2009.
- [12] G. Huston, "Interconnection, peering, and settlements," in *Internet Global Summit (INET)*, June 1999.
- [13] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Transactions on Networking*, 2001.
- [14] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 733–745, Dec. 2001.
- [15] J. Karlin, S. Forrest, and J. Rexford, "Autonomous security for autonomous systems," *Computer Networks*, Oct. 2008.
- [16] T. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, Apr. 2002.
- [17] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A prefix hijack alert system," in *Proc. USENIX Security Symposium*, 2006.
- [18] H. Levin, M. Schapira, and A. Zohar, "Interdomain routing and games," in *ACM STOC*, May 2008.
- [19] H. Chang, D. Dash, A. Perrig, and H. Zhang, "Modeling adoptability of secure BGP protocol," in *ACM SIGCOMM*, Sept. 2006.
- [20] Rensys Blog, "Pakistan hijacks YouTube." http://www.renysys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml.
- [21] Y. Liao, L. Gao, R. Guerin, and Z.-L. Zhang, "Inter-domain routing under diverse commercial agreements," *IEEE/ACM Transactions on Networking*, 2010.
- [22] J. Hastad, "Clique is hard to approximate to within $n^{1-\epsilon}$," *Acta Mathematica*, vol. 182, 1999.
- [23] L. Gao, T. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," *IEEE INFOCOM*, 2001.

APPENDIX

A. SIBLINGS

Because some of our results are based on CAIDA's AS graph [9], our model also includes *sibling* relationships, where two different ASes are owned by the same organization.

A.1 Modeling sibling relationships.

A recent paper [21] provides an excellent treatment of sibling-to-sibling relationships that we adapt for our purposes. First, our model of export policies must account for sibling relationships:

GR2s AS b only exports a path via AS c to AS a if at least one of a and c are customers or *siblings* of b .

Our **NE** export rule now uses the modified **GR2s** (see Section 2.2). Next, in addition to considering customer, peer, and provider paths, the work of [21] introduces two new path types:

Sibling down. The first edge(s) on the path are sibling edges, and the first non-sibling edge is a customer-provider edge. A path that contains exclusively sibling edges is also considered sibling down.

Sibling up. The first edge(s) on the path are sibling edges, and the first non-sibling edge is a peer-to-peer or provider-customer edge.

Our modified model of local preferences is also based on [21]:

LPs Prefer customer paths, over sibling down paths, over peer paths, over provider paths, over sibling-up paths.

As discussed in [21], captures a type of "hot potato routing", where the AS prefers to send traffic outside its organization rather than carrying it through its own network.

A.2 Sibling rivalry in CAIDA's AS graph.

Sibling-to-sibling relationships seem to be the grand 'fudge-factor' in works that involve AS-level business relationships. CAIDA is the first to acknowledge the challenges of dealing with sibling-to-sibling relationships [9]; their approach is based on manually assigning these relationships to two ASes if they are owned by the same organization. This means that a large AS (*e.g.*, AS1239, with almost 1400 customers) can be a sibling of a tiny AS (*e.g.*, AS1803, with only four customers) if the two are owned by the same organization (*e.g.*, Sprint). The problem this causes is best illustrated by an example.

Figure 15: We show CAIDA's snapshot of the local topology around AS 1239 and AS 1803. Because CAIDA classes AS1803 and AS1239 as siblings, our model suggest that tiny AS 1803 will carry traffic from

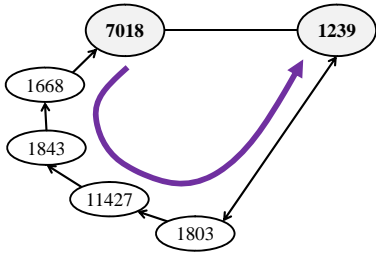


Figure 15: The trouble with siblings.

his provider AS 11427 to the large network of AS 1239; in fact, our model suggests that AT&T Worldnet’s Teir 1 AS 7018, that has over 2.2K customers, would route all traffic for AS 1239 over the long customer path through the sibling AS 1803. This is, of course, completely ridiculous. In practice, AS1803 is unlikely to advertise transit paths through AS1239 to any of it’s providers; AS 1239 essentially acts like a provider for AS1803, despite the fact that the two ASes are owned by the same organization.

To deal with these unbalanced sibling relationships, we preprocess CAIDA’s data as follows:

Sibling preprocessing: We convert sibling-to-sibling relationships to customer-provider relationships when at least one sibling has more than seven customers, and one sibling is at least twice the size of the other sibling.

This approach does remove some of the the artificially long paths we describe above. However, because CAIDA’s AS-relationship inference algorithms *starts* by using heuristics to assign sibling relationships, and then proceeds to infer the other relationships, we suspect that these sibling relationships can introduce inaccuracies in the results. On the other hand, these inaccuracies do not seem to matter very much, given that the results we obtained on the preprocessed CAIDA dataset matches well with the results we obtained from the Cyclops dataset that has no sibling edges (see Appendix H).

B. SIMULATION METHODOLOGY

We sketch the algorithms we developed for our simulations.

B.1 Routing tree algorithm.

At the core of our experiments is a *routing tree algorithm* that simulates the paths that each AS will choose to reach a prefix owned by a legitimate destination AS d . The routing tree algorithm assumes that ASes use the routing policies of Section 2.2, and is implemented using a specialized three-stage breadth-first search (BFS) on the AS graph.

1st stage. Our model of routing policies assumes that ASes prefer short path through their customers over all other paths; as such, we first construct a partial routing tree by performing a BFS ‘upwards’ from

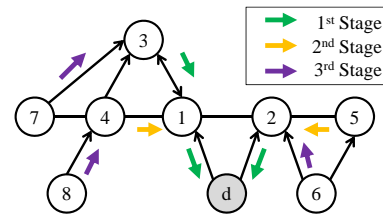


Figure 16: Routing tree algorithm.

the ‘root’ node d , using only customer-to-provider or sibling-to-sibling edges. If an AS is offered equal length paths through both a customer and a sibling, the BFS forces the node to choose the customer path. (In Figure 16, this amounts to adding edge $(d, 1)$ then $(d, 2)$ then $(1, 3)$).

2nd stage. Next, we capture the fact that (1) **GR2** allows only a single peer-to-peer edge to exist on any path through the network, and that (2) nodes prefer short paths through peers over paths through providers. To do this, in the second stage of the algorithm, we use only peer-to-peer edges to connect *new* nodes to the nodes *already added* to the partial routing tree in the 1st stage of the algorithm. (In Figure 16, this amounts to adding edges $(1, 4)$ and $(2, 5)$ but not $(1, 2)$ or $(7, 4)$).

3rd stage. Finally, we add provider/sibling up paths. We do this by traversing the existing partial routing tree with a BFS, and adding *new* nodes to the tree using provider-to-customer or sibling-to-sibling edges. (In Figure 16, this amounts to adding edges $(2, 6)$, $(3, 7)$ and finally $(4, 8)$ but not $(3, 4)$). Again, when an AS is offered equal length paths through both a provider and a sibling, the BFS forces the node to choose the provider path.

We capture **TB**, the fact that ASes break ties on AS numbers, by ensuring the that BFS traverses lexicographically by AS number. We capture **NE**, the fact that a node announces a path to *all* of his neighbors (except when forbidden by **GR2**), by running the algorithm above on all the edges in the AS graph.

B.2 Simulating the “Shortest-Path Export-All” attack strategy.

Given a (manipulator, victim) pair (m, d) , we use the routing tree algorithm to determine the outcome of each “Shortest-Path Export-All” attack strategy on each secure routing protocol as follows:

BGP. In this attack, both the manipulator m , and the legitimate destination d originate the IP prefix (See Section 3 and 4.1). To simulate this, we run the routing tree algorithm with *two* roots, m and d .

Origin Authentication/soBGP/S-BGP. Observe that this strategy requires the manipulator to announce, *to all his neighbors*, an attack path that is no longer than his shortest available path (see Section 3 and 4.1). We

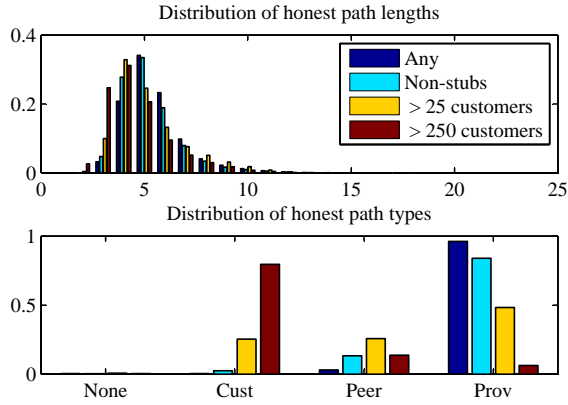


Figure 17: Path length and type distributions

simulate the “Shortest-Path Export-All” attack strategy using the following trick: First, we augment the AS graph with *fake nodes* corresponding to all the ASes on the manipulator’s attack path, excluding the manipulator and victim themselves. These fake nodes are given negative AS numbers. Then, we connect the victim to the manipulator via customer-provider edges through these fake nodes. Thus, the *fake path* is always the manipulator’s shortest customer path to victim, that is through an AS with lowest possible AS number (a negative number). Thus, our routing policies in Section 2.2 require the manipulator to choose this path. Thus, to simulate the “Shortest-Path Export-All” attack strategy, it suffices to run the routing tree algorithm on the AS graph augmented with the fake path.⁵

C. PATH DISTRIBUTION

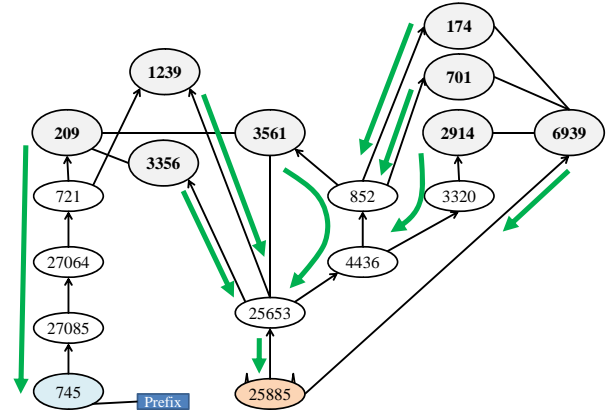
As a sanity check of our routing model, we show the distribution of path lengths and path types.

Figure 17: We show the distribution of path length and path type when all ASes behave normally. The distribution is over a randomly-chosen destination, and a source chosen from the same four classes as in Figure 4. We can see majority of paths in the Internet are short (about 5 hops on average), and further that larger ASes tend to have slightly shorter paths. Furthermore, as expected, we find that smaller ASes tend to use provider paths most frequently, while larger ASes tend to use customer paths most often, and that medium sized “Tier 2” ASes with at least 25 customers uses the largest (relative) fraction of peer paths.

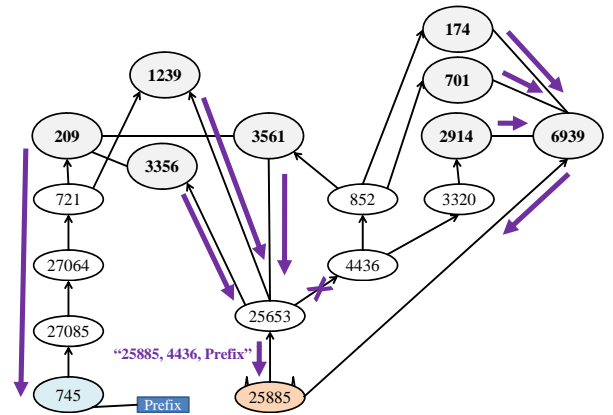
D. CLEVER FALSE LOOPS

We explain the example in Section 6.3 in more detail.

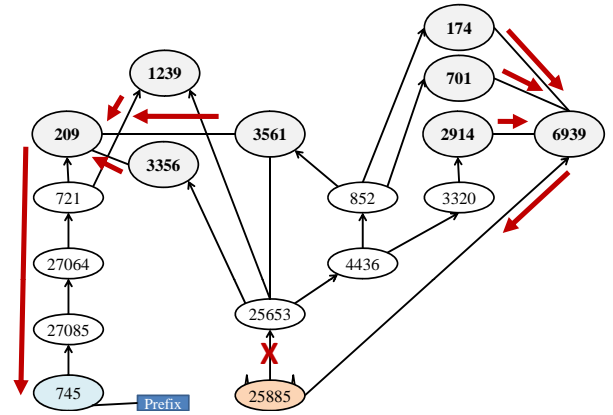
Figure 18(a). Simple Prefix Hijack. In the simple prefix hijack, the manipulator, AS 25885, the Bat



(a) Simple Prefix Hijack



(b) False Loop Prefix Hijack



(c) Prefix Hijack, Filtering Announcements to 25653

Figure 18: Using false loops.

⁵To account for BGP loop detection, we also include a simple check in the routing tree algorithm that cause a real node to reject a path that contains it fake counterpart.

Blue Corporation in Clifton, NJ, announces the path (28558, Prefix) to both of his providers, Fortress ITX’s AS 25653, a NJ-area ISP, and Hurricane Electric’s AS 6939, a large American backbone provider that is often considered to be a Tier 1 network. The manipulator manages to attract traffic from most of the Tier 1 ASes in the Internet, with the exception of QWest AS 209. However, many of these Tier 1’s, namely AS 3561, AS 174, AS 701 and AS 2914, use long, five-hop customer paths to the manipulator. The results of the attack is that the manipulator manages to blackhole traffic from a total of 32010 ASes.

Figure 18(b). False Loop Prefix Hijack. We now show how the manipulator can attract traffic from an additional 360 ASes by using a clever ‘false-loop prefix hijack’ attack. Now, the manipulator’s clever strategy is to announce the path (25885, Prefix) to his large provider AS 6939, while announcing the false loop (28558, 4436, Prefix) to his other provider AS 8220. As such, AS 4436 will no longer forward traffic to his customer Fortress ITX, AS 25653, choosing to forward traffic over an alternate peer path (not shown). Thus, the manipulator has eliminated a customer path from the network, and many of the Tier 1 ASes, including AS 3561, AS 174, AS 701, and AS 2914, will be forced to forward traffic over shorter peer paths. (Thus, AS 174, AS 701, and AS 2914 now use a three-hop peer path, instead of five-hop customer paths used in the simple prefix hijack.) These ASes now become more attractive to the rest of the Internet, increasing the volume of traffic flowing through the manipulator to 32370 ASes. Notice that the manipulator’s strategy ensures that Fortress ITX’s AS 25653 still forwards its traffic to the manipulator. Since quite a few Tier 1 ASes, namely Sprint’s AS 1239, Level 3 AS 3356, and Savis AS 3561, route through Fortress ITX’s AS 25653, the false loop prefix hijack strategy ensure that the manipulator does not lose a large amount of traffic by eliminating customer paths from the network.

Figure 18(c). Prefix Hijack and Filtering. Now suppose the manipulator decides to eliminate a customer path from the network by suppressing the announcement to Fortress ITX’s AS 25653. By doing this, the manipulator eliminates the customer path used by Sprint’s AS 1239 and Level 3’s AS 3356, as well as the peer path used by Savis AS 3561, and these Tier I ASes will now forward their traffic to the legitimate destination AS 745 instead. Thus, the manipulator loses traffic from about 2K ASes, attracting traffic from a total of 30028 ASes, and we find that the manipulator would have been better off if he had used the simple prefix hijack instead.

E. FAILED INTERCEPTION ATTACKS

We provide an example that proves the bottom-

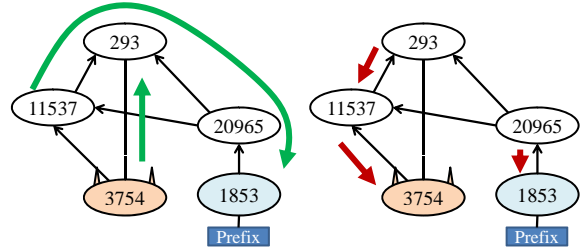


Figure 19: Disrupting a path through a peer.

middle and middle-right X entries in Table 1, and shows that there is an error in claims made in Section 2.2. of [1].

E.1 Export to provider, disrupt peer path.

We prove that the manipulator can lose a peer path to the victim by announcing an attractive path to his provider:

Figure 19: We consider an attack on BGP, where the manipulator falsely originates the victim prefix. The manipulator, AS 3754, is NYSERNet, a not-for-profit corporation that fosters science and education in New York State, while the victim, AS 1853, is the AConet Backbone, that provides services to multiple universities in Austria. The manipulator has a single provider, AS 11537, a single peer, AS 298, and 44 customers (not shown). The left (green) figure shows the normal outcome, where the manipulator has a paths to victim available through both his peer and his provider. The middle (red) figure shows what happens when the manipulator announces the victim’s prefix to his provider AS 11537; now, his peer AS 293 has *two* available customer paths of equal length. Since AS 11537 has a lower AS number than AS 20965, our **TB**rule requires AS 293 to choose the path through AS 11537 that leads to the manipulator. The manipulator has now “blackholed” himself; both his peer and his provider forward traffic to the manipulator, and none of the manipulator’s customers have any path to the victim AS 1853.

E.2 Export to peer, disrupt provider path.

We can also use the example of Figure 19, with a slight modification, to prove that the manipulator can lose a provider path to the victim by announcing an attractive path to his peer. Assume that AS 11537 has no customer or peer paths, nor any provider paths shorter than two hops, available to the victim AS 1853. In that case, if the manipulator *does* announce a path to his peer AS 293, but *not* to his provider AS 11537, the provider will prefer his two-hop provider path (293, 3754, Prefix) over any path to the legitimate victim, and again the manipulator creates a blackhole.

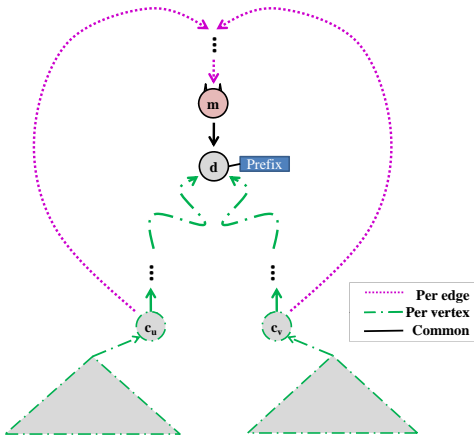


Figure 20: DILEMMA for proving hardness.

F. FINDING OPTIMAL ATTACKS IS HARD

We now show that, from the perspective of the manipulator, finding the optimal traffic attraction attack on BGP is computationally hard. We shall then show that, in fact, not only is finding the optimal attack hard, but even finding a “reasonable” attack, that is “not far” from the optimum, *i.e.*, approximates the optimum, is computationally hard. Our hardness results are obtained via a general proof technique that can be applied to show similar impossibility results for optimal (and approximate) traffic attraction attacks on other security enhancements to BGP (*e.g.*, SBGP, soBGP, and more).

We start by presenting our proof technique. We then show how it can be used to obtain hardness results for traffic attraction attacks on BGP; these results amount to showing that it is hard for the manipulator to decide which *paths* to export to which neighbors. We then move on to showing that even if the manipulator is restricted to announcing the normal paths (*e.g.*, because the network uses data-plane verification), that it is still hard for the manipulator to decide which *neighbors* to export to.

F.1 Key Ideas and Outline of Our Proofs.

The DILEMMA network. Our computational hardness results rely on showing the potential existence of the following scenario (see Fig. 14): The manipulator m is directly connected to the destination d . m wishes to attract as much traffic as possible, while all other nodes behave normally. The network contains two nodes, c_u and c_v , each with many direct and indirect customers whose routes to d go only through it. The number of nodes in the trees beneath c_u and c_v , that are of equal size, is significantly bigger than the number of nodes in the rest of the network. Hence, m ’s main goal is to attract c_u and c_v ’s traffic. However, in our constructions below, m shall always be able to attract *either* c_u ’s or c_v ’s traffic, but will be *unable* to attract *both* nodes’ traffic

simultaneously. Thus, m will have to choose which one of the two nodes to attract, inevitably losing the traffic of the other node and of all nodes in the subtree beneath it. m ’s inability to attract both c_u and c_v (alongside its ability to attract each of them alone) shall play a crucial role in our proofs.

Once we prove the existence of a small network as described above, that we term “DILEMMA”, we use it as a building block in a reduction from the MAX-INDEP-SET problem, that is a notoriously computationally hard problem.

The MAX-INDEP-SET problem. The MAX-INDEP-SET problem is defined as follows:

DEFINITION F.1 (INDEPENDENT SETS). Let $G = (V, E)$ be a graph. A subset of the vertices $I \subseteq V$ is an independent set if there is no edge in E between two vertices in I .

DEFINITION F.2 (MAX-INDEP-SET). In the MAX-INDEP-SET problem the input is a graph $G = (V, E)$ and the objective is to find an independent set I of maximum size.

The following is well known:

THEOREM F.3. MAX-INDEP-SET is NP-hard.

Reducing from MAX-INDEP-SET. We now outline our reductions from MAX-INDEP-SET to the problem finding an optimal attack on BGP (or security enhancements to BGP), that establish the computational intractability of the latter.

Given an instance of MAX-INDEP-SET $G = (V, E)$ we construct a network such that computing the traffic-attraction-maximizing attack in the network is equivalent to computing a maximum independent set in G . The node-set in our network contains the destination node d , the manipulator m , and a node c_v for each vertex $v \in V$ (and some additional nodes, as explained below). m is directly connected to d .

We ensure that, for each edge $e = (u, v) \in E$, m shall only be able to attract either c_u ’s or c_v ’s traffic, but not both nodes simultaneously, by constructing DILEMMA for c_u and c_v (adding nodes and links appropriately). Importantly, our constructions of DILEMMA gadgets are consistent, in the sense that if the manipulator cannot attract node c_v in one such gadget (because it chose to attract the other node in that gadget), then it also cannot attract c_v in all other DILEMMA gadgets that c_v participates in. Fig. 20 illustrates the vertex-specific, edge-specific, and general components of each DILEMMA construction (for each pair of neighboring nodes, c_u and c_v , that are connected by an edge (u, v) in E).

Now, consider an attack by m . Observe that because the trees beneath the c_v ’s constitute the vast majority of

the nodes in the network, and because the nodes in the tree beneath each of the c_v 's can only connect to d via that node, the success of m 's attack is measured by how many of the c_v 's it was able to attract. By construction, if two vertices in V , u and v , are connected by an edge in G then m cannot attract both c_u and c_v and thus the vertices corresponding to nodes that m is able to attract form an independent set in G . The converse is also true: Let $I \subseteq V$ be an independent set in G , then m can attract all the c_v 's corresponding to vertices in I (because no two such nodes participate in a DILEMMA construction).

Therefore, a maximum independent set in G corresponds to a traffic-attraction-maximizing attack in our network, and vice versa. The NP-hardness of MAX-INDEP-SET (and the fact that our reduction is computationally-efficient) now implies the NP-hardness of finding an optimal attack.

On the hardness of approximating the optimal attack. In fact, the close connections, presented above, between independent sets in G and traffic attraction, when combined with the following theorem, due to Hastad, imply a stronger result.

THEOREM F.4. [22] *Given a graph $G = (V, E)$, finding an independent set of size at least $\frac{OPT}{|V|^{\frac{1}{2}-\epsilon}}$, where OPT is the size of the maximum independent set in G , is NP-hard.*

Using the above theorem, and the exact same construction as before, we can now show that not only is finding the optimal attack computationally-hard, but so is finding an attack that approximates (in terms of number of attracted nodes) the optimal attack within any constant factor!

F.2 Finding Optimal Attacks on BGP is Hard!

We present the following theorems:

THEOREM F.5. *Finding an attack on BGP, that maximizes the traffic volume that goes through that node, is NP-hard.*

THEOREM F.6. *Finding an attack on BGP that approximates the optimal (traffic-volume-maximizing) attack within a constant factor C , is NP-hard for any constant C .*

PROOF SKETCH. The proofs of both theorems follows the outline presented in Sec. F.1. Hence, the main ingredient of the proof is showing the existence of a DILEMMA construction. We shall now present the DILEMMA construction; here, the manipulator's dilemma will be to decide *which path* should be announced to *which neighbors*. His strategy will be similar to the "false loop prefix hijack" of Section 6.3.

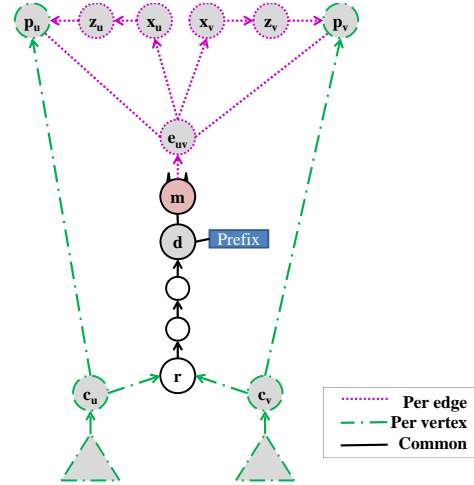


Figure 21: BAT-FROM-HELL-I.

The DILEMMA construction. Consider the network in Fig. 21, called "BAT-FROM-HELL-I". m is the node that wishes to attract as much traffic as possible for the victim prefix, while all other nodes behave normally. The network is such that

1. each of the nodes c_u and c_v has a large number of (direct and indirect) customers k in the subtree below it that can only reach d through it. Let k be big enough so that m be much more concerned with attracting c_u and/or c_v than with attracting all other nodes in the drawing;
2. p_u and p_v have lower AS numbers than r . Hence, if faced with a choice between the 4-hop route to d through r and a (false) 4-hop path to the prefix that has either p_u or p_v as next-hops, both c_u and c_v would prefer the latter route.

We now show that while m can attract c_u 's traffic, or c_v 's traffic, it cannot attract both nodes' traffic simultaneously. To see why this is true, consider node m 's options. Observe that for m to attract c_u 's (and its customers) traffic, it is necessary that c_u be offered a route of length 4 or less by p_u (because c_u already has an available route of length 4 through r). Recall that nodes prefer customer routes over peer routes (by **LP**) and so p_u prefers routes in which z_u is its next-hop node over routes in which the next-hop node is e_{uv} . Recall that when faced with two customer routes, they prioritize shorter routes (by **SP**). Unfortunately, observe that, no matter what m does, any route from p_u to m that has z_u as a next hop cannot be of length less than 4 (in fact, this is the case even if m hijacks d 's prefix and announces it to e_{uv}). Hence, if p_u routes through z_u then c_u 's available route through p_u shall consist of at least 5 hops and therefore will not be chosen by c_u .

How can m prevent p_u from routing through z_u ? The easiest way is, of course, simply not to announce a route

to e_{uv} . However, this will also mean that p_u will not learn of any route that goes through m . To avoid this, m must use a “false loop prefix hijack” strategy as in Section 6.3). He will announce a route to e_{uv} that contains one of the nodes x_u or z_u . By doing so m can ensure that one of these nodes shall not propagate this route further because of BGP’s loop detection mechanism, and that p_u still have a loop-free route through m that is announced to it directly by e_{uv} . For example, if m announces $mz_u d$ to e_{uv} then p_u learns the route $e_{uv}mz_u$ from e_{uv} and no route from z_u . Therefore, p_u shall make the route $p_u e_{uv}mz_u$ available to c_u , which, in turn, will choose this 4-hop route. Thus, m can attract c_u ’s traffic. Similarly, m can attract c_v ’s traffic by announcing the route mz_v to e_{uv} .

Can m attract both c_u and c_v at the same time? The answer is NO. Recall that to attract c_u m must include one of the nodes in the set $\{x_u, z_u\}$ in its announced route. Similarly, to attract c_v m must include one of the nodes in the set $\{x_v, z_v\}$. However, if m ’s announced route contains at least one node from each of these sets, and d , then p_u ’s route must be of length at least 4 and so both c_u and c_v shall not have a 4-hop route through p_u . This will result in both c_u and c_v choosing to forward traffic to r .

The reduction. We prove the correctness of the above two theorems via the arguments in Sec. F.1. We reduce from MAX-INDEP-SET. For every vertex $v \in V$ we create a node c_v . For every edge $e = (u, v) \in E$, we construct a BAT-FROM-HELL-I gadget to ensure that m not be able to attract both c_u and c_v simultaneously. Fig. 21 describes the construction of BAT-FROM-HELL-I for the edge (u, v) (illustrating the per-vertex, per-edge, and common to all gadgets, parts of the construction). Observe that our constructions of BAT-FROM-HELL-I gadgets are consistent, in the sense that if the manipulator cannot attract node c_v in one such gadget (because it chose to attract the other node in that gadget), then it also cannot attract c_v in all other BAT-FROM-HELL-I gadgets that c_v participates in. The arguments in Sec. F.1 now imply the theorems. \square

Extending to origin authentication and soBGP. The proof strategy above can easily be extended to attacks on origin authentication by adding more nodes and edges to the BAT-FROM-HELL-I. The modified BAT-FROM-HELL-I construction adds an extra node between r and d in Figure 21, and extra node y_i between nodes x_i and z_i with edges from y_i to nodes m and d . Then we use a similar argument as above to obtain the theorem.

F.3 It’s still hard, even if the manipulator must announce normal paths.

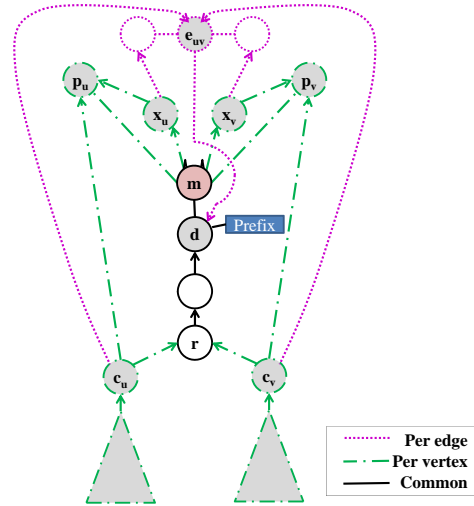


Figure 22: BAT-FROM-HELL-II.

The reader might suspect that the computational task shall become much easier if the manipulator is severely constrained by security mechanisms (and hence the space of feasible attacks it must consider is significantly smaller). Surprisingly, this is not the case. We show that the above results hold even if the security mechanism (*e.g.*, data-plane verification) forces the manipulator to announce his *normal path*! We show that finding the optimal attack is computationally hard even if the only decision the manipulator makes is *whether or not to export its normal path* (and thus, the path it actually uses).

THEOREM F.7. *Even if the manipulator may only announce the normal path, finding an attack that maximizes the traffic volume through the manipulator is NP-hard.*

THEOREM F.8. *Even if the manipulator may only announce the normal path, finding an attack that approximates the optimal (traffic-volume-maximizing) attack within a constant factor C , is NP-hard for any constant C .*

PROOF SKETCH. The proofs of both theorems follows the outline presented in Sec. F.1. Hence, the main ingredient of the proof is showing the existence of a DILEMMA construction. We shall now present such a construction.

The DILEMMA construction. Consider the network in Fig. 21, called “BAT-FROM-HELL-II”. m is the node that wishes to attract as much traffic as possible, while all other nodes are behaving normally. The network is such that

1. each of the nodes c_u and c_v has a large number of (direct and indirect) customers k in the subtree below it that can only reach d through it. Let k be big enough so that m be much more concerned

with attracting c_u and/or c_v than with attracting all other nodes in the drawing;

2. p_u and p_v have lower AS numbers than r . Hence, if faced with a choice between the 3-hop route to d through r and a 3-hop route to d that has either p_u or p_v as next-hops, both c_u and c_v would prefer the latter route.

We now show that while m can attract c_u 's traffic, or c_v 's traffic, it cannot attract both nodes' traffic simultaneously. To see why this is true, consider node m 's options. m is forced to announce its normal path to d , md . Hence, m 's only decision is to which neighboring nodes to announce the route md . Observe that if m announces md to x_u , then p_u will choose the customer route $p_u x_u md$ over the peer route $p_u md$ (by LP). This will result in c_u choosing the 3-hop route through r over the 4-hop route through p_u . Similarly, if m announces md to p_v this will result in the loss of c_v 's traffic. Therefore, to attract c_u 's traffic it is necessary that m not announce a route to x_u and, similarly, to attract c_u 's traffic it is necessary that m not announce a route to x_u . Observe that if m does not announce md to both x_u and x_v then the edge e_{uv} shall be forced to choose its only available (provider-learned) route to d , $e_{uv}d$. In this case, both c_u and c_v will have a v -hop route to d through e_{uv} (and will choose it by SP). This will result in m 's loss of both c_u 's and c_v 's traffic.

The above shows that while m can easily attract c_u 's traffic alone (by not announcing md to x_u and announcing md to all other neighbors), or c_v 's traffic alone (by not announcing md to x_v and announcing md to all other neighbors), it cannot attract both c_u and c_v 's traffic simultaneously.

The reduction. We prove the correctness of the above two theorems via the arguments in Sec. F.1. We reduce from MAX-INDEP-SET. For every vertex $v \in V$ we create a node c_v . For every edge $e = (u, v) \in E$, we construct a BAT-FROM-HELL-II gadget to ensure that m not be able to attract both c_u and c_v simultaneously. Fig. 22 describes the construction of BAT-FROM-HELL-II for the edge (u, v) (illustrating the per-vertex, per-edge, and common to all gadgets, parts of the construction). Observe that our constructions of BAT-FROM-HELL-II gadgets are consistent, in the sense that if the manipulator cannot attract node c_v in one such gadget (because it chose to attract the other node in that gadget), then it also cannot attract c_v in all other BAT-FROM-HELL-II gadgets that c_v participates in. The arguments in Sec. F.1 now imply the theorems. \square

F.4 Two Remarks

Attraction v.s. Interception. While our results are stated for attraction attacks (as they only discuss

the amount of traffic that the manipulator can attract), the fact that in all of our DILEMMA constructions the manipulator is directly connected to d , and so always has a route available, implies that all of our hardness results extend to interception attacks.

The degree of the manipulator. Our hardness results are in the number of edges that the manipulator has (that is roughly the size of V in the MAX-INDEP-SET instance). However, the result in Sec. F.2 can easily be made to hold even if the manipulator only has a constant (even 2) number of neighbors. This can be achieved via the addition of intermediate nodes. In contrast, our result in Sec. F.3, where the manipulator only chooses whether to announce its actual path to each neighbor, is computationally easy if the manipulator has a constant number of neighbors (as it can simply go over all the possibilities).

G. GUIDELINES FOR INTERCEPTION

We prove the results marked with a \checkmark in Table 1. That is, we provide guidelines that guarantee that a manipulator's attack strategy preserves an available path to the victim IP prefix.

To do this, we consider the *normal outcome*, where the all nodes behave normally, and the *manipulated outcome*, where a single AS m , the manipulator uses some attack strategy that *deviates* from the normal routing policies of Section 2.2. The victim IP prefix is legitimately owned by a destination AS d . Let the nodes on m 's available path to d in the normal outcome be a_1, \dots, a_{t-1} , so that m routes to d on the path $ma_1 \dots a_t$ (where for convenience we will set $d = a_t$). We would like to guarantee that the manipulator's attack strategy leaves him with an available path to d through a_1 (in the manipulated outcome). That is, we want to guarantee that a_1 will not route through m in the manipulated outcome.

G.1 A useful lemma.

Before we start, we need the following useful concept:

Transitive customers. A node b is a *strict transitive customer* of node c if b is connected to c via a path consisting of only customer-provider links as in the right half of Figure 23. We also restate here a simple, useful lemma of the Gao-Rexford conditions proved by Gao, Griffin and Rexford in [23].

LEMMA G.1 ([23, THEOREM VII.4]). *If either the path $P = abRc$ or the path $P' = cR'ba$ is available, and if node a is not a customer of node b , then node c is a strict transitive customer of node b over the available path.*

We remark that Lemma G.1 still holds as long as all the nodes on the available path (except perhaps the

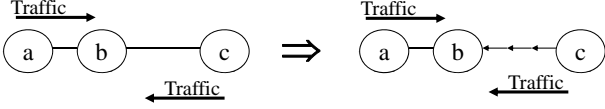


Figure 23: Lemma G.1.

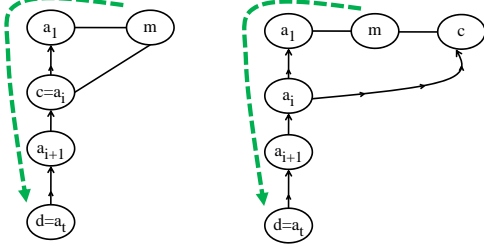


Figure 24: Case 1 (left) and Case 2 (right) in Claim G.2.

last one, closest to the destination) behave normally, according the routing policies in Section 2.2.

G.2 Available path through peers/customers. May export to peers & customers.

We prove the four results \checkmark^* results in the top left corner of Table 1. The following claim that does *not* require **GR1**:

CLAIM G.2. *Suppose that nodes use the routing policies of Section 2.2. Suppose m 's path to d in the normal outcome is a peer or customer path (i.e., a_1 is a peer or customer of m). Then m has an available path through a_1 in manipulated outcome, even if m announces any (possibly false) path to any of his neighboring peers or customers.*

PROOF. First, notice that if m 's available path in the normal outcome is a peer or customer path, then **GR2** tells us that a_1 's available path in the normal outcome must be a customer path, and Lemma G.1 immediately tells that for every $i \in [t-1]$, a_{i+1} is a customer of a_i . By **NE**, it follows that every a_i hears an announcement from his customer a_{i+1} .

Let c be any neighbor node of m that heard a path announcement from m . Recall that c must be either a peer or customer of m . We now have two cases:

- Suppose that c is one of the nodes on m 's available path in the normal outcome, i.e., $c = a_i$ for any $i \in [t-1]$. We argued above that a_i learns a path from its customer a_{i+1} . Now, recall that by definition m is a provider or peer of n . It follows from **LP** that for $c = a_i$, the customer path through a_{i+1} is more attractive than the peer or provider path through m , and so $c = a_i$ will prefer to route through a_{i+1} .

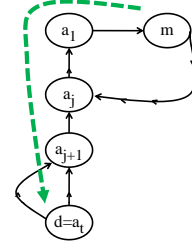


Figure 25: Proof of the induction step in Claim G.3.

- Suppose that n is not one of the nodes on m 's available path in the normal outcome. Repeatedly applying **GR2** tells us that the only nodes that can hear about c 's path through m must be strict transitive customer of c . Suppose that some a_i for $i \in [t-1]$ hears about the path through m . It follows that a_i learns about the path through m from his provider. Again, by **NE** a_i hears an announcement from his customer a_{i+1} , and by **LP** this customer path through a_{i+1} is preferred over the provider path through m .

It follows that in each case, every a_i for $i \in [t-1]$ will prefer to route through a_{i+1} instead of routing through m . In particular a_1 has a path to d that does not go through m . By **NE**, a_1 will announce this path to m and the claim follows. \square

G.3 Available path through customers. May export to providers.

In Section E.1, we presented an example that proves that if a_1 is peer of m , then m may lose an available path through a_1 by lying to one of his neighboring providers. However, we now prove the \checkmark in the top right of Table 1, showing that if a_1 is a customer of m , then m can even get away with lying to his neighboring providers. This claim requires **GR1**:

CLAIM G.3. *Suppose that **GR1** holds, and that nodes use the routing policies of Section 2.2. Suppose m 's path to d in the normal outcome is a customer path (i.e., a_1 is a customer of m). Then m has a available path through a_1 in the manipulated outcome, even if m announces any (possibly false) path to any of its neighbors.*

PROOF. Now, observe that if a_1 's available path to d in the manipulated outcome is unchanged, then by **NE** a_1 announces this path to m and we are done. Thus, we suppose that the path $a_1 \dots a_t d$ is not used in the manipulated outcome. It follows that there must be some node a_i for $i \in t$ that is closest to the destination d that forwards traffic over a different path in the manipulated outcome (i.e., different from the $a_i \dots a_t$ path he used in

the normal outcome). The proof now follows from the following (backward) induction from $j = 1 \dots i$.

Base case. Let $a_{j+1} = a_{i+1}$. From the way we defined a_i , it follows that a_{i+1} uses the same customer path to d in the normal outcome and the manipulated outcome, so it follows that a_{i+1} 's available path does not go through m .

Induction step. Suppose that in the manipulated outcome a_{j+1} uses a customer path to d that does not go through m . Then in the manipulated outcome a_j also forwards along a customer path to d that does not go through m .

We now prove the induction step. First, observe that the Lemma G.1 and the fact that a_1 uses a customer path in the normal outcome immediately tells us that a_{j+1} is a customer of a_j . By **NE**, a_{j+1} must export a path to a_j in the manipulated outcome; thus, a_j has a customer path available in the manipulated outcome. By **LP**, it follows that whatever path a_j chooses in the manipulated outcome must also be a customer path. To finish the proof of the induction step, we shall show, by contradiction, that this path does not go through m : Suppose that the available path that a_j chooses in the manipulated outcome goes through m . Then, since this path is a customer path, Lemma G.1 tells us that the manipulator m as a strict transitive customer of a_j along this path. Now recall that that m uses a customer path in normal outcome, and apply Lemma G.1 again to obtain that that a_j must be a strict transitive customer of m . It follows that there is a customer-provider loop in the AS-graph (between a_j and m), which violates **GR1**, and we have arrived at our contradiction.

From the induction, we learn that a_1 must use a customer path in the manipulated outcome that does not go through m . By **NE**, a_1 announces this path to m and the claim follows. \square

G.4 Available path through providers. May export to customers.

We showed how a manipulator might disrupt an available path through a *provider* by announcing to a *provider* (Section 5.1), or a *peer* (Appendix E.2). We now show that a manipulator that wants to preserve an available path through a provider *may* export any path to his *customers*, proving the \checkmark on the bottom left of Table 1. We again rely on **GR1**:

CLAIM G.4. *Suppose that **GR1** holds, and that nodes use the routing policies of Section 2.2. Suppose m 's path to d in the normal outcome is a provider path (i.e., a_1 is a provider of m). Then m has a path available through a_1 in the manipulated outcome, even if m announces any (possibly false) path to any of its neighboring customers.*

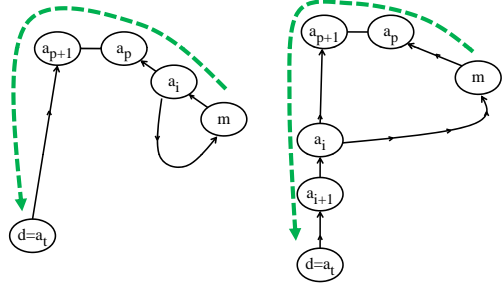


Figure 26: Case 1 (left) and Case 2 (right) in Claim G.4.

PROOF. Since m only announces paths to his customers, repeated applications of **GR2** immediately tell us that the only nodes that can hear about paths through m are strict transitive customer of m . Now consider m 's available path $a_1 \dots a_t$, and let a_p be the node closest to m such that m is a strict transitive customer of a_p . (We know that a_p exists since in particular a_1 , a provider of m , is one such node.) We now show that no a_i will choose to route through m :

- Suppose some node a_i for $i \in [p]$ learns about the path through m . We argued above that a_i must be a strict transitive customer of m . However, by the definition of a_p , m is also a strict transitive customer of a_i ! It follows that there is a customer-provider loop in the AS graph, which violates **GR1**. It follows that no a_i for $i \in [p]$ will learn about the path through m .
- Suppose some node a_i for $i = p \dots t - 1$ learns about the path through m . Above we argued that a_i must be a strict transitive customer of m , so it follows that a_i learns about the path through m from his provider.

Now, by the definition of a_p and **GR2** we know that a_{p+1} is either a peer or customer of a_p . Applying **GR2** again tells us that a_{i+1} is a customer of a_i for each $i = p+1 \dots t-1$. By **NE**, we know that a_{i+1} announces a path to a_i for every $i = p \dots t-1$. It follows that for every a_i for $i = p \dots t-1$, the path it learns through its peer or customer a_{i+1} is more attractive than the provider-path through m .

It follows that in each case, every a_i for $i \in [t-1]$ will route through a_{i+1} instead of routing through m . In particular a_1 will have a path to d that does not go through m . By **NE**, a_1 will announce this path to m and the claim follows. \square

H. CYCLOPS+IXP DATASET

This appendix presents versions of all the graphs in this paper, computed from the ‘Cyclop+IXP’ AS Graph datasets [10, 11]. We constructed this dataset from the November 20, 2009 Cyclops dataset, by removing 276 edges connected to 4-byte ASNs, and removing 444 edges with unclassified business relationships. Then, we augmented the dataset with 21890 peer-to-peer edges from the recent IXP dataset [11], using only edges with good confidence, and ignoring edges that referred to ASes that were not in the Cyclops dataset. We note that the Cyclops dataset does *not* include any sibling-to-sibling edges, and is also derived using a different relationship inference algorithm than the CAIDA dataset.

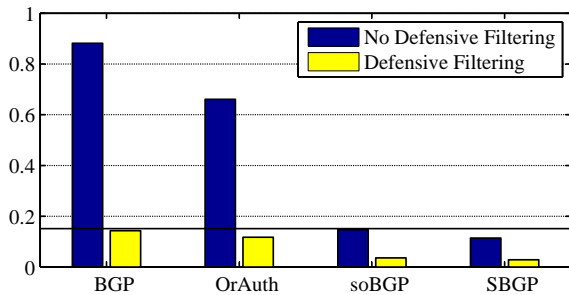


Figure 27: Lower bounds on the probability of attracting at least 10% of ASes in the Internet. Cyclops+IXP dataset.

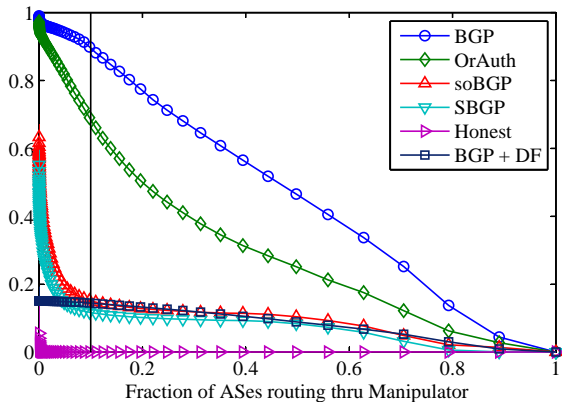


Figure 28: CCDF for the “Shortest-Path Export-All” attack strategy. Cyclops+IXP dataset.

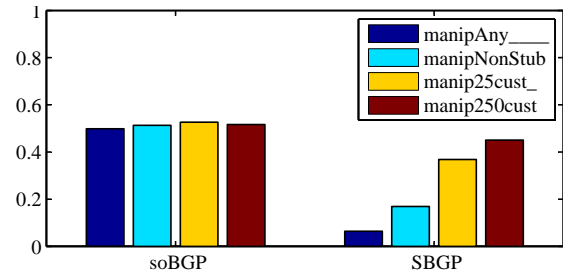


Figure 29: Probability of finding a shorter path. Cyclops+IXP dataset.

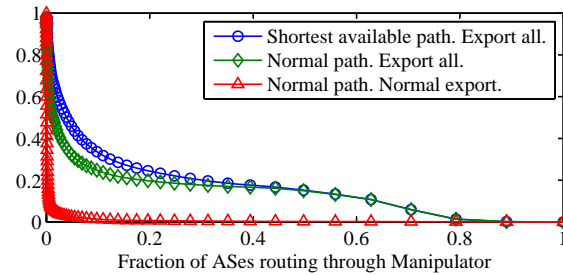


Figure 30: Aggressive export policies. Cyclops+IXP dataset.

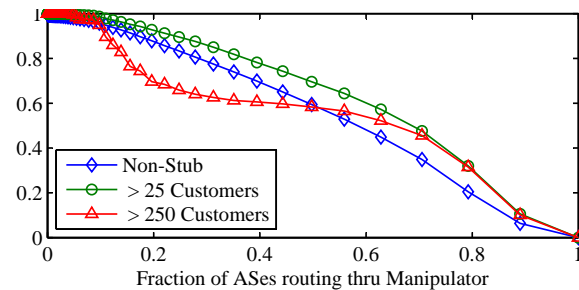


Figure 31: “Shortest-Path Export-All” attack strategy on BGP by different manipulators. Cyclops+IXP dataset.

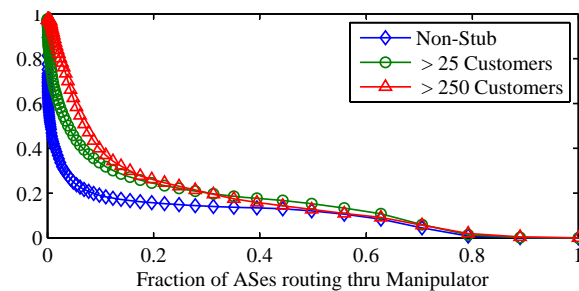


Figure 32: “Shortest-Path Export-All” attack strategy on S-BGP/data-plane verification by different manipulators. Cyclops+IXP dataset.

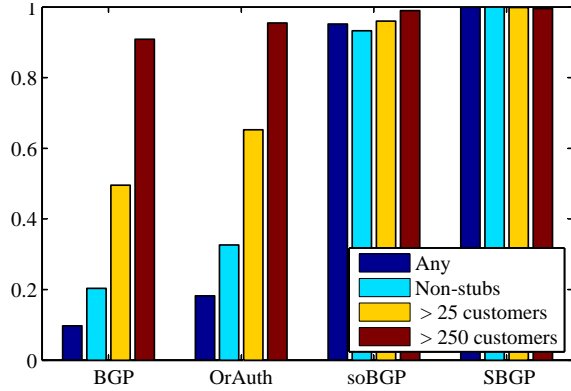


Figure 33: Probability that the “Shortest-Path Export-All” attack strategy does *not* create a blackhole. Cyclops+IXP dataset.

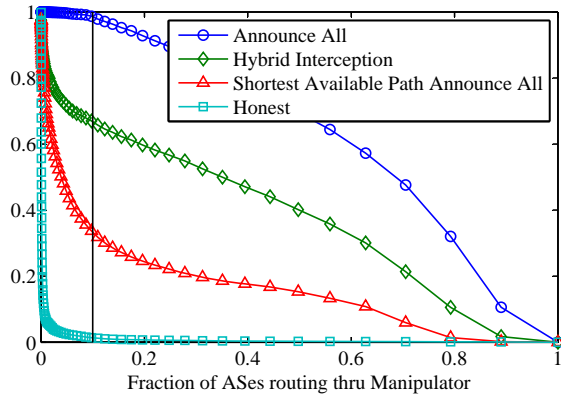


Figure 34: Interception attacks on BGP. Cyclops+IXP dataset.

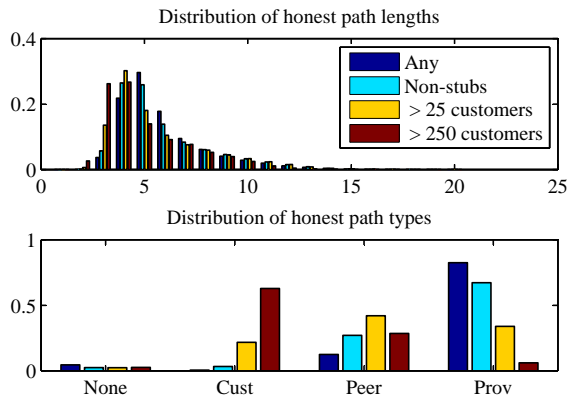


Figure 35: Path length and type distributions. Cyclops+IXP dataset.