You Can't Handle the Lie: Next-Hop Verification in BGP

Clay Thomas
claytont@cs.princeton.edu

Gavriel Hirsch
gbhirsch@cs.princeton.edu

January 11, 2019

Abstract

This paper presents a new protocol called *Next-Hop Verification*, which that reduces the set of contexts in which other autonomous systems are incentivized to lie while participating in BGP. The protocol works by sharing information about BGP path announcements between different ASs, using the existing structure of the network, and checking those path announcements against the true flow of traffic in the data plane. We discuss the advantages and disadvantages of this new approach, and compare its effectiveness to that of previously considered verification techniques.

1 Introduction

1.1 Background and Previous Work

Routing on the Internet involves many distinct Autonomous Systems (AS's), each with its own data sources, destinations, and links; as well as its own preferences over how traffic is routed. An AS may prefer that the traffic it sends and receives be sent over the shortest path, in order to decrease latency; or it may prefer to send its traffic through or avoiding specific other AS's for economic incentives, due to contracts between AS's about routing costs; or it may prefer to avoid certain other AS's, if it is concerned about malicious activity. In the other direction, an AS may also prefer to attract or deter traffic from certain other AS's, again for economic incentives or perhaps even to spy on certain traffic.

These AS's typically use the Border Gateway Protocol (BGP) to announce routes to neighbors and learn routes from neighbors in the control plane, and to then choose how to actually route traffic in the data plane. However, there is no way for BGP to enforce the requirement that an AS route traffic in a way that matches its announcements. Thus, due to all of the various (often conflicting) preferences that AS's have over how traffic is routed, these AS's can often have incentives to lie in the control plane about what they will actually do in the data plane.

To counteract this, *verification protocols* have been developed which run alongside or alter BGP in order to prevent lying. Unfortunately, directly verifying the routes a packet takes in the data plane requires cryptographic signatures on every packet, as in [PS03]. This huge overhead makes data-plane verification impractical. Instead, previous work has searched for control pane protocols which still manage to prevent or discourage lies.

Much work has been done on analyzing BGP verification protocols through game-theoretic models. [LSZ08] shows that in a general set of contexts, a form of verification called *path verification*[1] ensures that no AS or group of AS's can get strictly preferred routes for its traffic by telling lies.

However, lying can potentially give other benefits beyond getting better routes for your own traffic. [GHJ+08] analyzes BGP games in which agent's utility may depend on attracting traffic from other AS's. In this scenario, even using path verification does not suffice to disincentivize lying. They introduce another form of verification called *loop verification*, which is simpler but weaker, and describe conditions under which path or loop verification do disincentivize lying. However, they admit that many of these conditions are unreasonably strong, such as requiring that AS's always announce either all paths they are aware of or none at all to all of their neighbors.

There is also much discussion of convergence in BGP. [LSZ08] argues that if we assume that the network infrastructure is not changing over time and that each AS makes BGP announcements based solely on a ranking of paths that is also constant over time, then subject to a condition called *No Dispute Wheels* the network will converge to a stable set of routes. In more general contexts though, convergence becomes very hard to reason about. Thus, in this paper we choose to focus on what happens after convergence. We show that if the network were to converge to a state that depends on lies, we would then be able to catch the lies and shame the liar. As a result, it should not be beneficial to lie in a way that leads to that state.

Not sure what to do with this paragraph.

1.2 Our contributions

Given the difficulty of preventing lies in BGP, we would at least like to be able to detect those lies. The initial observation of this work is that there will always be at least one AS that knows what each other AS is truly doing, namely the one that directly receives traffic from it in the data plane. As a result, if AS's are willing to collaborate then there is information that can be used to detect the existence of lies, without requiring full data-plane verification. Based on this observation, we present a new protocol called *next-hop verification* and show that it effectively prevents lies in certain scenarios.

As with previous practical verification protocols, next-hop verification protocol runs in the control plane. However, it also uses information sampled from the data plane in order to aid verification. Specifically, it requires AS's to keep track of which of its neighbors forward traffic towards it for different destinations. Given that agents have this information, next-hop verification gives an effective way for agents to distribute queries and fact-checking using the existing structure of the network and no encryption.

We find that next-hop verification allows us to catch lies assuming there are is no traffic attraction among preferences. In this regard, next-hop verification is similar in effectiveness to path verification. In the context where preferences involve traffic attraction, we are able to significantly weaken the assumptions which [GHJ+08] needed on the preferences of AS's. In this sense, next-hop verification is sometimes more powerful than path verification (although there do exist cases where path verification will prevent a lie that next-hop verification cannot). Additionally, we find that next-hop verification is strictly more powerful than loop verification, that is, every lying situation detected by loop verification will be detected by next-hop verification. In general, we attempt to embark on a similar program to that of [GHJ+08], experimenting with various settings and seeing where next-hop verification leads to good incentive properties.

Our general analysis of next-hop verification does however use the strong assumption that there

I've not thought about collusion-proofness for next-hop. Unless we do, I'd significantly shorten this paragraph

[1] In the original paper they refer to it as route verification.

is only a single lying agent. We also for simplicity and power focus on the situation in which everyone else participates fully. That said, the protocol would still provide some value even with only partial participation, and we will discuss this in Section 5.

Finally, it is worth noting that next-hop verification is "bulkier" than loop verification, as it essentially has to distribute information across the whole collection of AS's, as well as do some minimal data-plane monitoring.

work this in nicer

1.3 Organization

In the rest of Section 1 we discuss existing work, and we briefly outline our results and their limitations. Section 2 informally presents the model we use. Section 3 describes the next-hop verification protocol. Section 4 states and proves some theorems about the implications of using next-hop verification, and goes through examples of concrete scenarios to compare what would be possible with and without next-hop verification. Finally, Section 5 offers some conclusions and suggestions for follow-up work.

Might not need this in a "workshop style paper"

2 Model Details

2.1 BGP framework

We model the network of AS's as an undirected graph, with a node for each AS and an edge between any two AS's that can directly communicate with each other without going through a third AS. We assume that the graph is a single connected component, so any AS can in theory interact with any other AS (although in practice, say if an intermediate AS intentionally drops traffic, this may not always actually be possible). As is standard in the literature, we assume there is a unique destination AS d, because routing to different destination (prefixes) is done independently in BGP.

what is this comment for?

In the BGP framework, AS's can announce the existence or removal of paths to each other. Each AS has an import policy that determines how it responds to path announcements from neighboring AS's. Specifically, the import policy determines whether the AS will update its route, given a new announcement. ((PERHAPS DISCUSS THE NOTION OF PREFERENCES AND THUS MENTION HONESTY HERE))

In this paper we ignore any concerns of storage for keeping track of all announcements from neighbors, so we assume that any newly observed path is added to a table, unless the receiving AS is already in the path. In this case it will either ignore the path announcement, or if the AS never announced the subpath containing itself it will raise an alarm that another AS has exported a false path. We also assume that on hearing an announcement, the AS can only take actions related to the full path, and not related to particular subpaths.

what is this sentence for?

It's too early to talk about look verification

what does this mean?

Each AS also has an export policy which determines how it will communicate the paths it is aware of to other AS's. Specifically, for a BGP compliant AS, the export policy determines whether or not to announce the AS's currently chosen route to a given neighbor. In realistic settings an AS may prefer not to announce its path to all neighbors (the most famous example of this being [GR01], in which for example customers will not route traffic between two of their providers). However, for a manipulative, non-BGP compliant AS, a much richer set of export policies is available. The AS can lie arbitrarily, announcing path that it doesn't use or that don't even exist.

Finally, each AS has some preferences over how the actual traffic in the network flows. The AS's will choose a strategy, namely their import and export policies, based on these preferences.

We assume that the collection of strategies leads the network to converge to a stable solution. See [LSZ08, GHJ+08, GSW02, GSW99] for examples of more formal details about proving different types of convergence and what assumptions are necessary.

Definition 1. *We say that an AS* ***m*** *is* ***lying*** *if the stablility of the post-convergence equilibrium depends on a neighbor thinking that* ***m*** *is acting in a way that is inconsistent with the true equilibrium. We say that an AS is* ***honest*** *if it is not lying.*

This is how I would make this definition

Definition 2. *Suppose a BGP network G has reached a stable equilibrium. We say that an AS* ***m*** *is* ***lying*** *if the it exports a route other than that which it is using. We say that an AS is* ***honest*** *if it is not lying.*

2.2 Verification

In this section, we give an overview of some previously considered verification protocols. These will be our comparison points for next-hop verification.

Definition 3. *In a network using* ***path verification*** *it is impossible for AS's to announce that they are using paths which were not already announced to them.*

Some extensions to BGP, such as S-BGP, can enforce path verification. However, it requires additional overhead as well as universal adoption [LGS13], because every route communicated in the control plane must be cryptographically signed by every AS along that route.

Definition 4. *In a network using* ***loop verification*** *no AS will use an export policy that involves not sending a path to a neighbor specifically because that neighbor is already in the path. In addition, if an AS ever sees a path containing itself that it did not announce, it will "raise an alarm", with the idea that the offender (the first node which announced a false path) can be publicly shamed.*

Note that if instead export policies did not send paths to neighbors who are already in them, the alarming in loop verification could not always be done.

Since loop verification is very minimal and easy to adopt we will assume that the network uses it.

We don't want to do this. Next-hop encompassing loop is an interesting theorem.

2.3 Behavioral assumptions

When agents get utility from attracting traffic, stronger verification protocols are needed to disincentivize lying. In [GHJ+08], the authors consider the following classes of traffic attraction.

Definition 5. *An AS m has* ***generic attraction*** *if it gets utility depending on the paths of agents which route through m. More specifically, m gets utility given by a function of the collection of routes containing m which are used by other nodes.*

In particular, in generic attraction an AS m has incentives to effect *how* other agents route through m. For example, a provider may want a customer to route directly through it in order to charge that customer more.

Volume attraction, on the other hand, is more restrictive:

Definition 6. *An AS m has* ***generic attraction*** *if it gets utility depending on which agents route through m. More specifically, m gets utility given by a function of the collection of agents whose route contains m.*

Volume attraction reflects truly malicious situations such as spying, where the manipulative agent wants to view packets for some nefarious reason. ((POSSIBLY MENTION EG CHINA SPYING ATTRACTION FIASCO)).

((DEFINE NEXT-HOP POLICY MAYBE))

3 Next-Hop Verification Protocol

We now define next-hop verification. This protocol is run on an already-stabilized BGP network, i.e. it starts after convergence of BGP has already occurred. Nodes communicate along existing links in the network, storing and sending next-hop queries. We assume that AS's can "raise the alarm", similar to [GHJ+08], which refers to alerting other agents that something is going wrong with the particular query. For our results section, we will assume that when the alarm is raised, the offending agent will be caught (for example, via the collaboration of the NANOG mailing list to detect the problem, as suggested in [GHJ+08]).

Each node maintains a queue of queries which it needs to answer. A query is denoted $Q_d(a, b)$, representing a node announcing that a uses b as its next-hop in its path to destination d.

```
Denote the acting AS by n
function INITIALIZE
    for each hop (a, b) in n's path to d do
        Add the query Q_d(a, b) to your query queue
function RESPOND(Q_d(a, b))
    if n previously responded to Q_d(a, b) then
        return
    if n = a then
        if n does not use b as its next hop for d then
            "raise the alarm"
        return
    if n = b then
        if a does not use n as its next hop for d then
            "raise the alarm"
        send the query Q_d(a, b) to all neighbors
    else (i.e. n ≠ a, b)
        if a uses n as its next hop for d then
            "raise the alarm"
        else
            send the query Q_d(a, b) to all neighbors
function MAIN
    for each query Q_d(a, b) in queue do
        RESPOND(Q_d(a, b))
    clear the queue
```

3.1 Additional notes

- In the case where n is responding to a query $Q(a, n, d)$, it needs to check whether a actually forwards traffic directly to n for destination d, which must be done in the data plane. Accordingly, each AS should keep a flag for each other (neighboring AS) $\times$ (dest AS) pair, representing whether the first AS ever directly sends n traffic destined for the second AS.
- If an AS has too many neighbors and/or destinations and keeping all these flags becomes unmanageable, each AS can have a policy for determining which pairs it thinks are important to monitor for.

4 Results

The following lemma is the key to our positive results. It explains whey next-hop verification

Lemma 1. *Let G be a BGP instance with the next-hop verification phase. Suppose all nodes except one manipulator m are next-hop participants. Assume that m announces to node v a hop (a, b), where in the data plane the hop (a, c) is used for some $b \neq c$. (Note that both b and c may be used if $a = m$ and m is "faking traffic"). Furthermore, suppose there exists a path from v to c not containing m. Then m with be caught by next-hop verification, and will receive utility $-\infty$.*

Proof. Let P denote the path from v to c not containing m. Because the activation sequence is fair, every node along the path with be activated, in the proper order going from v to c. Node v starts with the query $Q(a, b)$, and thus it will eventually travel to b, which will "raise the alarm" and give m utility $-\infty$. □

Note: should probably get rid of the protocol's ability to "drop queries" in view of a false-negative

Our first result formalizes the statement "in a network without traffic attraction, next-hop verification will catch all incentivized lies".

Theorem 1. *Let G be a stable outcome of a BGP instance without traffic attraction. Suppose there is a single manipulator m, and all other nodes honestly participate in BGP and in next-hop verification. If m lies in order to get a better path to d, then next-hop verification will catch that lie and shame m.*

Proof. Suppose m announces a hop (a, b) to v, where (a, c) is actually in a route from m to d for some $c \neq a$. For contradiction, assume that every path from c to v includes m (so m can drop the next-hop queries and its lie won't be caught). Now, the route from m to d which includes (a, c) must be a simple path, so it cannot include m. If there was a path from v to d which did not contain m, then the path from c to d would give a simple path from v to c. Thus, every route from v to d includes m.

Because all non-m nodes are honest, m must get v to change its route to d in order to get a different path by lying to v. However, v's route cannot effect m's route to d, because every route from v to d includes m. This contradicts the assumption that m got a better path by lying, and shows that there exists a path from c to v not including m. By the previous lemma, this means m will be caught and shamed. □

The next result shows that next-hop verification is at least as powerful as loop verification:

Theorem 2. *Let G be a stable outcome of a BGP instance. Suppose there is a single manipulator m who lies in the stable outcome, and all other nodes honestly participate in BGP and in next-hop verification. Then if m would be caught if loop verification were used by all other nodes, then m will be caught by next-hop verification.*

Proof. TODO: this □

As an aside, we note that an analogue of the above theorem for path verification does not hold[2]. Combined with the extensive discussion in [GHJ+08], the previous result gives us a few different situations in which next-hop verification will catch all incentivized lies. The rest of this section is dedicated to weakening the requirements for those incentive-compatibility properties to hold.

The following theorem says that next-hop verification will still catch lies in networks with volume attraction.

Theorem 3. *Let G be a stable outcome of a BGP instance with traffic volume attraction. Suppose there is a single manipulator m, and all other nodes honestly participate in BGP and in next-hop verification. If m lies in order to attract traffic from some node u, then next-hop verification will catch that lie and shame m.*

Proof. Suppose m did manage to attract more traffic from a victim u. Let P denotes the path u originally took to d, and let Q denote the path u takes in the manipulated outcome G. Note that $m \in Q$ but $m \notin P$. Let v denote a node that m lied to, and suppose v is told that hop (a, b) is used while (a, c) is actually used for $c \neq b$.

Because the lie told to v must effect the path chosen by u, there exists a path R from u to v not including m. Furthermore, because m uses the route S from c to d (and profits from it) we know $m \notin S$. Thus, by eliminating any possible loops from RPS, we get a simple path from v to c which does not include m. Thus, by the lemma, m will be caught and shamed. □

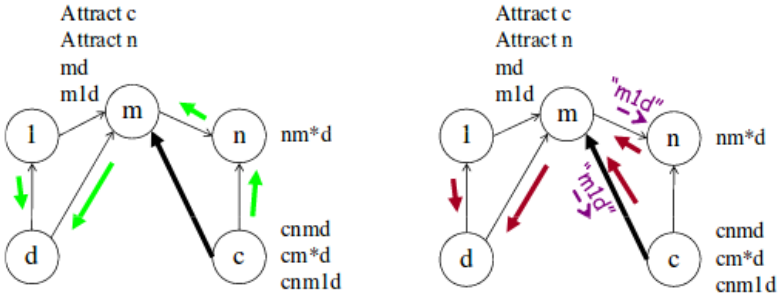
However, we are not able to extend this result to generic attraction. Indeed, consider the example given by (Figure 1), from [GHJ+08]. In this network, the manipulator m wants nodes n and c to use m as their next hop for destination d, for example, because of economic considerations. The AS's who know what m is doing in the data plane are separated from the AS's m is lying to. Those victim agents cannot communicate with d and l without going through m, which will just throw their next-hop queries away.

Conjecture 1. *Let G be a stable outcome of a BGP instance with generic traffic attraction. Suppose there is a single manipulator m, and all other nodes honestly participate in BGP and in next-hop verification. Furthermore, assume all nodes use next-hop policy in ranking their paths. If m lies in order to attract traffic from some node u, then next-hop verification will catch that lie and shame m.*

((NOTE: the right assumption in the above might be some sort of "next-hop attraction", which sounds similar to AT4 from Goldberg. However, discussing AT4 takes us way too far down the Gao-Rexford rabbit hole)).

[2]For an example, consider figure 1, altered by removing the link between nodes m and l. The exact same lying strategy is still available to m under next-hop verification, but this is not possible if path verification is used.

Figure 1: Bowtie



5 Possible future directions

We showed in the previous section that next-hop verification has the potential to provide some major benefits for catching lies in BGP, and thus reducing or even eliminating ASs' incentives to do so. However, there is still some future work that should be done before it is used in practice.

One important question is how effective the protocol can be in partial deployment or participation. In our theorems we assumed that there was only one malicious AS and that all others actively participated fully in helping to catch lies. However in practice, it maybe the case that some AS's have not deployed the necessary software and/or hardware for participation. It may also be the case that some AS's choose not to participate or to only share a limited subset of the information they have, even if they are not themselves malicious, lying agents.

Intuitively, it seems like partial deployment and participation would still be at least somewhat valuable. While this would not give us the same degree of benefits as full participation, it would still allow for the sharing of some information and thus preventing some lies. That said, it would be worth doing a more formal analysis of what this would look like.

Somewhat relatedly, our protocol uses an unencrypted "flooding" sort of approach to send queries through the network to their desired recipients. As we saw in Figure 1 though, if there is a bottleneck where all paths between two particular nodes goes through a malicious AS, it may want to read the queries that pass through it and drop ones that could expose it. More generally, in a situation with only partial deployment or with multiple malicious AS's, this problem could become more severe, cutting off AS's that both want to fact-check each other.

One possible way to resolve this would be to add encryption so that an intercepting AS cannot read the query. Or if there are concerns about the AS even knowing whether a query has been sent, the query could even be routed as though it was just normal traffic.

We see a possible value in using this sort of approach, but we also see benefits in the unencrypted "flooding" approach. The biggest problem we see with this alternative approach is that it makes it more complicated for AS's to adopt and deploy systems that can do next-hop verification, since it would require cryptographic techniques that they AS's may not already be using. If next-hop verification is ever deployed in practice, we think it is important to consider this and other tradeoffs between keeping it as simple as possible, and keeping it as secure as possible.

6 Conclusion

References

[GHJ+08] Sharon Goldberg, Shai Halevi, Aaron D. Jaggard, Vijay Ramachandran, and Rebecca N. Wright. Rationality and traffic attraction: Incentives for honest path announcements in bgp. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 267–278, New York, NY, USA, 2008. ACM.

[GR01] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, December 2001.

[GSW99] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. Policy disputes in path-vector protocols. In *Proceedings of the Seventh Annual International Conference on Network Protocols*, ICNP '99. IEEE, 1999.

[GSW02] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.*, 10(2):232–243, April 2002.

[LGS13] Robert Lychev, Sharon Goldberg, and Michael Schapira. BGP security in partial deployment: Is the juice worth the squeeze? *CoRR*, abs/1307.2690, 2013.

[LSZ08] Hagay Levin, Michael Schapira, and Aviv Zohar. Interdomain routing and games. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 57–66, New York, NY, USA, 2008. ACM.

[PS03] Venkata N. Padmanabhan and Daniel R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Comput. Commun. Rev.*, 33(1):77–82, January 2003.