Analyzing Internet Routing Security Using Model Checking

Adi Sosnovich [1], Orna Grumberg [1], and Gabi Nakibly [2]

[1] Computer Science Department, Technion, Haifa, Israel
{sadisos,orna}@cs.technion.ac.il
[2] National Research and Simulation Center, Rafael, Haifa, Israel
gabin@rafael.co.il

Abstract. The goal of this work is to enhance Internet security by applying formal analysis of traffic attraction attacks on the BGP routing protocol. BGP is the sole protocol used throughout the Internet for inter-domain routing, hence its importance. In attraction attacks an attacker sends false routing advertisements to gain attraction of extra traffic in order to increase its revenue from customers, drop, tamper, or snoop on the packets. Such attacks are most common on the inter-domain routing.
We use model checking to perform exhaustive search for attraction attacks on BGP. This requires substantial reductions due to scalability issues of the entire Internet topology. Therefore, we propose static methods to identify and automatically reduce Internet fragments of interest, prior to using model checking.
We developed a method, called BGP-SA, for BGP Security Analysis, which extracts and reduces fragments from the Internet. In order to apply model checking, we model the BGP protocol and also model an attacker with predefined capabilities. Our specifications allow to reveal different types of attraction attacks. Using a model checking tool we identify attacks as well as show that certain attraction scenarios are impossible on the Internet under the modeled attacker capabilities.

1 Introduction

In this work we combine static examination and model checking to examine fragments of the Internet and either identify possible attacks on their routing protocol or prove that specific attacks are not possible.

The Internet is composed of Autonomous Systems (ASes). Each AS is administered by a single entity (such as an Internet service provider, or an enterprise) and it may include dozens to many thousands of networks and routers. Inter-domain routing determines through which ASes packets will traverse. Routing on this level is handled throughout the Internet by a single routing protocol called the *Border Gateway Protocol* [16] (BGP).

It is well known that the Internet is vulnerable to traffic attacks [9,4]. In such attacks malicious Autonomous Systems manipulate BGP routing advertisements in order to attract traffic to, or through, their AS networks. Attracting extra traffic enables the AS to increase revenue from customers, drop, tamper, or snoop on the packets. In the recent past, there have been frequent occurrences of traffic attraction attacks on the Internet [21,20,18,19,12,13]. Some of those attacks allowed oppressive governments to block their citizens from accessing certain websites. In other attacks the perpetrators eavesdropped or altered the communications of others, while in different attacks spammers sent millions of emails from IP addresses they do not own. In one type of attack scenario the traffic is diverted through the attacker's AS network and then forwarded to its real destination, which allows the attacker to become a "man-in-the-middle" between the source of the traffic and its final destination. Such attacks are called *interception attacks*. In another type of attack scenario, the traffic is not forwarded to its real destination, which allows the attacker to impersonate the real destination or simply block access to it. Such attacks are called *attraction attacks*. In the sequel, when we refer to any attack of these types we call it a *traffic attack*.

Our goal is to provide insights to where and how BGP traffic attacks are possible. Note that BGP is the sole protocol used throughout the Internet for inter-domain routing, hence its importance. We develop a method that exploits model checking to systematically reveal BGP traffic attacks on the Internet, or prove their absence under certain conditions. Our method is based on powerful reductions and abstractions that allow model checking to explore relatively small fragments of the Internet, yet obtain relevant results. Reductions are essential as the Internet nowadays includes roughly $50,000$ ASes.

A fragment includes a destination and an attacker AS nodes. The goal of the attacker is to attract traffic sent to the destination while the goal of normal nodes is to direct the traffic to the destination.

In a normal mode of the BGP operation, when no attacker is present, an AS node receives from some of its neighbors their choice of routing path to the destination. When AS A announces a routing update to its neighbor AS B consisting of a target node n and a path π, it means that A announces to B that it is willing to carry packets destined to n from B, and that packets will traverse over the path π. From the announced routing paths, the node chooses its most preferred route (according to business relationship between the entities that administer the ASes, length of path, etc.) and sends it further to some of its neighbors. Its announced path may, in turn, influence the choice of preferred paths of its neighbors. In contrast, an attacker may send its neighbors faulty routing paths whose goal is to convince them and other AS nodes in the Internet to route through the attacker on their way to the destination.

Our static examination investigates the announcements flowing throughout the Internet. The basic idea is that if announcements cannot flow from one part of the Internet to another then nodes in the first part cannot influence the routing decisions of nodes in the second part. Our first reduction is thus based on BGP policies that determine the flow of announcements in the Internet. Given a destination and an attacker, we *statically* identify on the full Internet topology a *self-contained fragment* S that consists of a set of nodes, including the destination and attacker. S is defined so that nodes in S may send announcements inside and outside of S, but nodes outside of S never send announcements to nodes in S. Thus, the routing choices of nodes in S are not influenced by routing choices of the rest of the Internet.

We can now isolate S from the rest of the Internet and apply model checking only to it in order to search for an attack strategy that attracts traffic to the attacker. Since routing decisions in S are made autonomically, an attack strategy found on S will attract the same nodes from S when the full Internet is considered. This result allows to significantly reduce the processing burden on model checking while searching for attacks on the Internet. Similarly, if we show that no attack strategy manages to attract traffic from certain victims in S then the attacker will not manage to attract traffic from those victims in the full Internet as well. Thus, by searching a small fragment we find attacks on the full Internet or show their absence.

The second reduction we suggest is applied within a self-contained fragment S to further reduce it. We *statically* identify nodes in S that for all BGP runs choose the same route to the destination (that does not pass through the attacker), regardless of the attacker's behavior. Such nodes are considered *safe* with respect to the destination and the attacker of S.

The advantage of this reduction is twofold. First, safe nodes can be safely removed from the model, thus easing the burden on model checking. Second, nodes that wish to improve their routing security may decide to route through safe nodes, thus avoiding traffic attacks from this specific attacker. We further elaborate on the latter in Section 8.

Our third reduction is based on an abstraction. We can *statically* identify a *routing-preserving* set of nodes that all make the same routing choices. Such a set can be replaced by a single node with similar behavior without changing routing decisions of other nodes in the network.

Note that all three reductions are computed statically by investigating the Internet topology and are therefore easy to compute.

We implemented our method, called BGP-SA, for BGP Security Analysis. We first extracted from the Internet self-contained fragments, which are defined by a destination and an attacker nodes, and applied reductions to them. We chose the attacker and the destination nodes either arbitrarily or in order to reconstruct known recent attacks. In order to apply model checking, we modeled the BGP protocol for each AS node. We also modeled an attacker with predefined capabilities. The BGP model is written in C. We considered several specifications which allow to reveal different types of attacks. We ran IBM's model checking tool ExpliSAT [7] on self-contained, reduced fragments.

We found interception attacks. One of those attacks reconstructs a recent known attack where Syria attracted traffic destined to YouTube [18]. In other cases we showed that some attraction scenarios are impossible under the modeled attacker capabilities. In the latter case, model checking could also reveal additional *safe* nodes.

To summarize, the contributions of this paper are:

- Defining and implementing aggressive *reductions* of the Internet. The resulting (relatively small) self-contained fragments enable an automatic analysis.
- Developing an *automatic analysis* that can reveal possible attacks on the Internet and prove that certain attacks are not possible.
- Identifying *safe nodes* that are not amenable to traffic attacks and can be exploited to reduce vulnerability of other nodes in the Internet.

2 Related work

There are some past works that use formal methods to analyze convergence properties of BGP. [3] uses a static model of BGP path selection and analyzes configurations of BGP policy. [2] uses static and dynamic models to reason about BGP convergence. [17] analyzes convergence of routing policies with an SMT solver. We use a different modeling to reason about traffic attraction scenarios on the Internet. Our modeling implements runs of the protocol until stabilization, includes an attacker, and is based on the routing policy used by most ASes on the Internet. Our model includes parts of BGP that are most relevant to the analysis of traffic attraction, and is based on the model presented in [9].

Ref. [9] discusses the security of BGP and its vulnerability to different attacks. It shows that an attacker may employ non-trivial and non-intuitive attack strategies in order to maximize its gain. This was shown by giving anecdotal evidence (obtained manually) for each attack strategy in specific parts of the Internet. In our work we develop reductions and use model checking to systematically and automatically search for BGP traffic attacks on the Internet.

3 BGP background

The routers and networks of the Internet are clustered into connected sets. Each such set is called an autonomous system (AS). As of the end of 2014, there are roughly 50,000 autonomous systems on the Internet. An AS is usually administered by a single network operator, such as an ISP (Internet service provider), an enterprise, a university, etc. Each AS has a predefined routing policy determined by the network operator. An autonomous system is assigned a globally unique number, sometimes called an Autonomous System Number (ASN).

Routing of data packets on the Internet works in two levels:

1. Inter-domain routing that determines through which ASes the packets will traverse. This level of routing is handled by a single routing protocol called the Border Gateway Protocol [16] (BGP).
2. Intra-domain routing that determines the path taken by the packets within each AS. This is determined independently in each AS. Each network operator is free to choose any routing protocol to employ within its AS. The most common examples of such routing protocols are OSPF [15], RIP [14], or IS-IS [6].

Note that BGP is the sole protocol used for inter-domain routing. In essence, BGP is the glue that holds the Internet together and which allows to connect between different ASes. The currently used version of BGP is number 4. The protocol's standard is specified by the IETF (Internet Engineering Task Force) standardization body in [16]. The primary function of BGP is to exchange network reachability information between different ASes. Each AS periodically announces to all its neighboring ASes (i.e., the ASes to which it is directly connected) routing updates. A routing update consists of the identity of a target network and a path that consists of a sequence of ASes that starts from the advertising AS and leads to the AS in which the target network resides. Note that BGP advertises routing updates pertaining to networks residing within ASes (not to ASes themselves), while the routing path is at the AS level. When AS A advertises a routing update to its neighbor AS B consisting of a target network n and a path π, it means that A announces to B that it is willing to carry packets destined to n from B, and that packets will traverse over the path π. This routing information will then be propagated by AS B to its neighbors, after prepending itself to π. The propagation of routing information by one AS to all its neighbors is a matter of a policy determined by that AS. We shall elaborate on this in the following.

Every AS stores the routing updates learned from its neighboring ASes in a data structure called Adj-RIBs-In. If several routes were advertised for the same target network by different neighboring ASes, then the AS must choose its most preferable one. Once a route is chosen all packets destined to that target network will be routed via the neighboring AS that announced the chosen route. The chosen routes for all target networks on the Internet are stored in a data structure called Loc-RIB. Choosing the most preferable route is a matter of policy specific to each AS. In this paper we call it a *preference policy*.

As noted above, each AS propagates to its neighbors the routing updates it receives. Only routes within the Loc-RIB may be propagated. Namely, an AS can only propagate a route it has chosen as its most preferable one. Before propagating a route the AS must prepend itself to that route. An AS may choose a subset of its neighbors to which a route is propagated. This is a matter of policy specific to each AS. We call it an *export policy*.

Preference and Export Policies As noted above, the preference and export policies are a local matter for each AS determined by the network operator. These policies usually abide by business relationships and commercial agreements between the different network operators. While in reality there are many types of business relationships and agreements, the following two relationships are widely believed to capture the majority of the economic relationships [8].

- Customer-provider – in such a relationship the customer pays the provider for connectivity. Usually, the provider AS is larger and better connected than the customer AS. For example, the AS administered by Sprint is a provider of the AS of Xerox corporation. Xerox pays money to Sprint for connecting Xerox to the rest of the Internet through Sprint. In this paper we denote this kind of relationship with arrow from customer to provider.
- Peer-peer – in such a relationship the two peer ASes agree to transit each others traffic at no cost. Usually, the two ASes are of comparable size and connectivity. For example, the ASes administered by Sprint and NTT are peers. Each provides the other connectivity to parts of the Internet it may not have access to. In this paper we denote this kind of relationship with an undirected line between the two ASes.

Based on the above business relationships the following is a well-accepted model for the preference and export policies [8].

Preference Policy. This policy is based on the following simple rationale. 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). Combined with the fact that routing must be loop free and preferably on short routes the following policy is defined:

1. Reject a routing update that contains a route if the AS itself already appears on the announced route.
2. Prefer routes that were announced by a customer over routes announced by a peer over routes announced by a provider.
3. Among the most preferable routes choose the shortest ones, i.e., the ones which traverse the fewest ASes.
4. If there are multiple such paths, choose the one that was announced by the AS with the lowest ASN.

Export Policy. This policy is based on the following simple rationale. An AS is willing to carry traffic to or from other ASes only if it gets paid to do so. Based on this rationale the following policy is defined:

- AS B will announce to AS A a route via AS C if and only if at least one of A and C are customers of B.

To illustrate the above policies consider the topology depicted in Figure 1. Let us consider the routing of AS 9 to AS 0. There are three possible paths: (9,3,2,1,0), (9,4,5,0), and (9,7,1,0). Due to the above preference policy 9 will favor the first route over the second route which is favored over the third route. This is because the first route is announced by a customer AS (i.e., 3), while the second and third routes are announced by a peer (4) and provider (7) ASes, respectively. Note that the chosen route (9,3,2,1,0) will be propagated to 7 and 4, according to the above export policy.

4 BGP Modeling and Specifications

In this paper we use a BGP standard model acceptable in the literature [9] to facilitate the analysis of traffic attacks using false route advertisements. The model includes all the relevant parts of the protocol that deal with the dissemination and processing of route advertisements. In particular, the mechanisms of route distribution and route preference are modeled, including malicious routes originated by an attacker.

We assume a single destination, called $Dest$, such that the other ASes want to send traffic to a target network within $Dest$. We can focus on a single destination because routing announcements referring to different destinations flow independently of each other. Namely, the routing to one destination does not influence the routing to another destination. As a result, in our model a routing update does not include the identity of the target network.

Modeling the BGP Network A BGP network N is a tuple $N = (Nodes, Links, Dest, Attacker)$ where $Nodes$ is a set of Autonomous System (AS) nodes in the network graph. $Links$ is a set of node pairs with one of the following types: customer-provider or peer-to-peer, representing the business relationships between ASes in the network. $Dest$ is an AS from $Nodes$ representing a single destination node that contains the target network to which all other nodes build routing paths. $Attacker$ is a node from $Nodes$ representing an AS that can send false routing advertisements to achieve traffic attraction or interception.

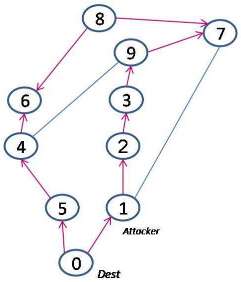

Fig. 1. BGP network example

$Dest$ and the $Attacker$ are called the *originators* of N. All other nodes are called *regular* nodes.

Consider the BGP network presented in Figure 1. $Nodes = \{0, 1, \ldots 9\}$, $Links$ consists of customer-provider links such as $(1 \to 2)$ and $(9 \to 7)$, and also peer-to-peer links such as $(4 - 9)$ and $(1 - 7)$.

A *path* in N is a sequence $\pi = (n_1, \ldots, n_k)$ of nodes in $Nodes$, such that for every $1 \leq i < k$, n_i and n_{i+1} are connected by an edge (of any kind) from $Links$.

Local states and Global Configurations The *local state* of a regular AS n consists of:

- A message queue $Q(n)$ containing incoming route announcements.
- A Routing Information Base $RIB(n)$ containing a set of possible routes to $Dest$. The most preferred route is denoted $chosen(n)$.

A (global) *configuration* of N consists of the local states of all nodes.

Next we define a run of the BGP protocol on network N. A run starts from an *initial* configuration in which all queues and RIBs are empty. Initially $Dest$ sends announcements to all its neighbors. The run terminates after all nodes in N terminate their run and their queues are empty. In particular, the originators have already sent out all their announcements. The final configuration of a run is called *stable*.

We often will be interested in referring to export actions along a run. We denote by $export(n, n')$ the action of node n exporting an announcement to its neighbor n'.

4.1 Attack definitions and specifications

Attacker goal The goal of the attacker in our model is to achieve traffic attraction or interception. We say that a node n is *attracted* by the attacker if in the stable configuration $chosen(n)$ is a path on which the attacker appears. A node n is *intercepted* by the attacker if it is attracted, and in addition the attacker has a routing path to the destination.

Successful attack A successful attack is a BGP run such that its final stable configuration satisfies the attacker goal. The attack strategy can be represented by the sequence of actions preformed by the attacker during the attack, where each of its action contains the sent announcement and a set of neighbors to which it was sent.

Normal outcome is the final routing choices of all ASes in N when the attacker acts like a regular AS.

Trivial attack strategy In the trivial strategy the attacker sends a false advertisement to all its neighbors and announces that the target network is located within its own AS.

Specifications To measure how successful a traffic attraction or interception attack is, we suggest specifications that compare the result of the attack to the normal outcome of the protocol run and to the result of the trivial attack, when applicable. We define specifications of *traffic attraction or interception* from any victim as follows: if the attacker can attract (or intercept) traffic from any victim, while it fails to do so in the normal run and the trivial attack, the attraction (or interception) specification is satisfied. We demonstrate how the specification is implemented in our model on Section 6.3.

5 Reductions and abstractions

The goal of our reductions is to obtain a manageable sized fragment of the large network which is suitable for identifying BGP traffic attacks or show their absence.

5.1 Self-contained fragments

The extraction of a self-contained fragment is our main reduction that significantly reduces the initial network, such as the full Internet topology. The reduction is based on preserving the flow of announcements in the network during a BGP run. The following is a central notion in our analysis of the flow. It directly follows from the export policy (see Section 3). A path $\pi = (n_1, \dots, n_k)$ in N is *valid* if n_1 is an originating node, no node is repeated on π, and for every $1 < i < k$, at least one of n_{i-1} and n_{i+1} is a customer of n_i. Further, no n_i is an originating node except n_1 and possibly n_k. Examples of valid paths in network N of Figure 1 are $(0, 5, 4, 6, 8)$ and $(0, 5, 4, 9, 3, 2, 1)$. Note that $(0, 5, 4, 6, 8, 7)$ is not a valid path, since both 6 and 7 are not customers of 8. The following is a key observation about valid paths.

Lemma 1. *If there is no valid path in N with edge from node n to node n' then there is no run in N along which $export(n, n')$ is performed.*

Note, however, that the contrary is not true. There might be an edge (n, n') on a valid path but still no $export(n, n')$ is performed. This is due to the preference policy of nodes.

We say that n *cannot export* to n' if there is no run in which the action $export(n, n')$ is performed.

Let N be a network and let $S \subseteq Nodes$ be a subset of its nodes that includes all originators of N. S is a *self-contained fragment* of N if for every $n \in (Nodes \setminus S)$, n cannot export to any $n' \in S$. This means that nodes outside of S cannot change routing decisions of nodes in S.
The following lemma describes the significance of self-contained fragments.

Lemma 2. *Let N be a network and let S be a self-contained fragment of N. Then, any traffic attack found on S can occur on N as well. Moreover, if we prove that a traffic attack is not possible in S then the corresponding attack is not possible in N as well.*

The lemma implies that instead of searching a huge network N (such as the Internet) we can identify a (relatively small) self-contained fragment, isolate it from the rest of the network, and search for possible attacks on it. Assume an attacker (in S) can attract traffic from a node n' in S. Then since nodes outside of S do not send n' alternative routing options, they cannot "convince" n' to change its routing choice and avoid the route through the attacker. Thus, a traffic attack which is successful in S is also successful in N. Similarly, if a certain node is definitely *not* routing through the attacker in S then the same holds in N as well.

Fragment importance Following Lemma 2 , it should be noted that the fragment concept is of great importance for applying significant reductions on BGP networks. The set of announcements that a node within the fragment can receive during any BGP run with an arbitrary attacker on the whole Internet is equal to its counterpart on a similar run that is applied to the fragment only. Therefore, the set of chosen routing paths within the fragment is equal as well, due to the deterministic preference policy of each node. Thus, the task of applying model checking on the whole Internet is reduced to applying it on a self-contained fragment when searching for BGP traffic attacks with our suggested specifications. Additionally, the fragment concept may be useful for other BGP-based formal analyses that require substantial reductions on large networks.

Computing self-contained fragments Given a network $N = (Nodes, Links, Dest, Attacker)$, we describe the computation of a set of nodes which forms a self-contained fragment. The resulting S includes $Dest$ and $Attacker$ and excludes some of N's nodes that cannot export any announcement to S.

Initially, only the set of originators $O = \{Dest, Attacker\}$ and their neighbors are in S. A node c outside of S is inserted to S if c is a neighbor of some $n \in S$, and c is on a valid path from some originator in O to n. The algorithm terminates when for every $c \notin S$ which is a neighbor of some $n \in S$, c is not on a valid path from an originator to n and therefore (by Lemma 1) c cannot export to n.

Example for a self-contained fragment extraction Consider the 10-nodes-sized network, presented in part A of Figure 2. In practice the initial network can be much larger. Applying the fragment extraction algorithm results in:

1. Initialization: Insert O and their neighbors. $S = \{22561, 48685\} \cup \{209, 25934, 6677\}$
2. Add $c = 3257$, due to path from $o = 22561$ to $n = 6677$: $(22561, 209, 3257, 6677)$
3. Add $c = 5580$, due to path from $o = 22561$ to $n = 25934$:$(22561, 209, 5580, 25934)$

The remaining nodes are not added. For example, 3303 does not appear on any valid path in the original network, and is therefore dropped during the construction of a self-contained fragment. After applying this phase we remain with 7 nodes as presented in part B of Figure 2.

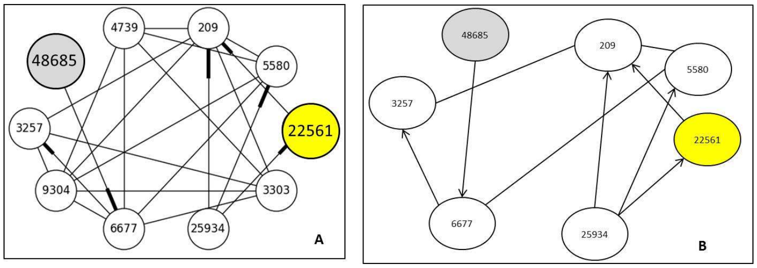

Fig. 2. Fragment example. The grey node 48685 is the attacker. The yellow node 22561 is the destination. The thick lines in part A represent the arrow direction of the customer-provider links.

5.2 Definite routing choice

In this reduction we identify nodes that never route via the attacker. If for all runs of BGP on a network N, a node n chooses to route through a specific path π originated by $Dest$ that does not pass through the attacker, then π is the *definite routing choice* of n, denoted $drc(n)$. We consider such nodes as *safe*, since they cannot be attracted by the attacker.

For example, in Figure 1, $drc(5) = (0)$ and $drc(4) = (5, 0)$. Node 5 is a neighbor of $Dest$ and its link to $Dest$ is more preferred than its other link. Therefore, since the announcement from $Dest$ is guaranteed to be sent to 5, it will always prefer this path regardless of other paths it might get from 4. For a similar reason, and since 5 is guaranteed to export its path to 4, node 4 will always prefer the route via 5. On the other hand, $drc(9)$ is undefined since on different runs its choice of routing may change as a result of the announcements sent by the attacker (which may change from run to run).

$drc(n)$, when defined, is $chosen(n)$ in every run, regardless of the attacker's actions. Consequently, the export actions of n are also determined. We can therefore eliminate such n from our network and initiate a BGP run from a configuration in which the results of its export is already in the queues of the appropriate neighbors. This may significantly reduce the network size to which model checking is applied.

5.3 Routing-preserving path

Another source of reduction is the abstraction of routing-preserving paths. A path $\pi = (n_1, \ldots, n_k)$ is *routing-preserving* if for every run r of N, in the final (stable) configuration of r one of the two cases holds: either for all $1 < i \leq k$, n_i chooses to route through n_{i-1} , or for all $1 \leq i < k$, n_i chooses to route through n_{i+1}.

Intuitively, for every run of the protocol, the nodes on a routing-preserving path all agree on the same route to the destination. As a result, we can replace such a path with a single node (an *abstraction* of the path) without changing the routing of other nodes in the network. The protocol of an abstract node is adjusted such that it exports announcements with lengths that match the number of nodes in the path it represents. An example of a routing-preserving path in Figure 1 is $(2, 3, 9)$.

6 The BGP-SA method

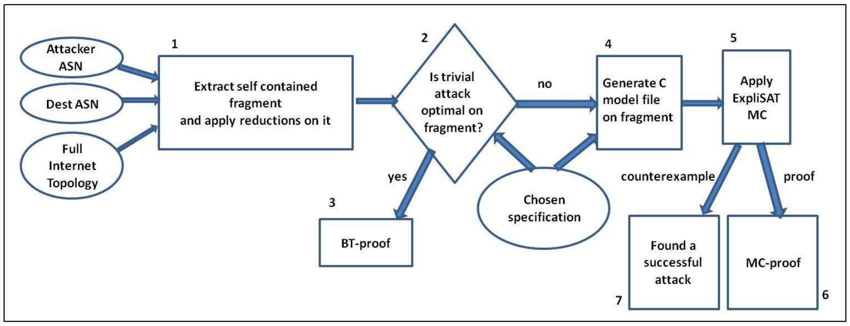
Attacker ASN
Dest ASN
Full Internet Topology
1
Extract self contained fragment and apply reductions on it
2
Is trivial attack optimal on fragment?
no
yes
3
BT-proof
Chosen specification
4
Generate C model file on fragment
5
Apply ExpliSAT MC
counterexample
proof
Found a successful attack
7
MC-proof
6

Fig. 3. The BGP-SA method

Our suggested method, called BGP-SA, for *BGP Security Analysis* uses reductions and model checking to apply a formal analysis of BGP attraction attacks on a large network topology. We use model checking to perform a systematic search for traffic attacks. A systematic search is essential in order to reveal non-trivial attraction strategies on topologies from the Internet. It has a major advantage over simple testing techniques that randomly search for attacks. The model checker we use can perform full verification, thus it can also prove that no traffic attack is possible under certain conditions.

The BGP-SA method is composed of several stages, as depicted in Figure 3. Below we describe them in details.

6.1 Reducing the network topology

The input to the BGP-SA method consists of the full network topology, the chosen attacker and destination ASN, and the chosen specification. Given this input, we first extract a self-contained fragment and apply additional reductions and abstractions. (see square 1 of Figure 3). The extraction and reduction algorithms are explained in Section 5. The output is a reduced fragment that contains the nodes within the extracted fragment S, without those for which drc is defined. (See Figure 4).

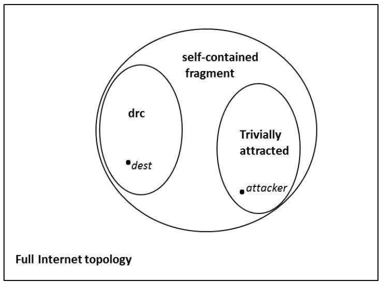

Fig. 4. Partition of node types in the extracted fragment

6.2 Simulating the trivial attack

Here we explain items 2-3 of Figure 3. Given a reduced fragment, we run a simulation of the trivial attack on it. If the chosen specification is traffic attraction and if all the nodes in the reduced fragment are trivially attracted, then the attacker cannot improve its attraction results. If the chosen specification is traffic interception and if the trivial attack satisfies the interception condition additionally to attracting all nodes in the reduced fragment, then again the attacker cannot improve its attraction results.

In both cases it is considered a proof (denoted *BT-proof* for Best Trivial attraction proof) that within the fragment the attacker does not have a strategy which is better than the trivial one. When BT-proof is obtained, the analysis is terminated and model checking is not needed. Otherwise, the nodes of interest for searching attraction scenarios are the remaining nodes that are neither trivially attracted nor have a defined drc, as presented in Figure 4.

6.3 Generating the C model

Given the reduced fragment and the chosen specification, we generate a model written in C on which the analysis is applied (see square 4 of Figure 3). Code 1.1- 1.3 depicts a pseudo-code of the generated code in high level, and below we give more details of it.

- Code 1.1 describes the procedures that implement nodes in our model. AS_Proc is the procedure of a regular AS. Its path preference and export policy are as explained in Section 3. The attacker has two procedures: $Arbitrary\_Attacker\_Proc$ is the procedure of an attacker that originates arbitrary path announcements and sends them to arbitrary neighbors. $Trivial\_Attacker\_Proc$ is the procedure of an attacker that applies the trivial attack and announces itself as the destination to all its neighbors. $Dest\_Proc$ is the procedure of $Dest$, in which it announces itself as the destination to all its neighbors.
- Code 1.2 describes the function implementing a BGP run in our model. The input parameter of this function is the type of run: normal – where the attacker acts as a regular AS, trivial – where the attacker applies the trivial attack, or arbitrary - where the attacker acts arbitrarily. The function is composed of a loop, where at each loop iteration each one of the AS procedures is activated once. A stable state is achieved when no message is sent by any AS and all the queues are empty. Convergence is guaranteed [11] due to the routing policies that are used in the model and the finite number of announcements that can be sent by the attacker. We bound the number of announcements originated by the attacker by letting it export to each neighbor at most one announcement. The function returns the routing results at the stable state which include $chosen(n)$ for each node n in the netwrok, where $chosen(n)$ is the preferred route of n.
- Code 1.3 describes the main function in the model and the assertion statement that implements the specification. The main function is composed of three calls to the function BGP_run, with the three types of run: normal, trivial, and arbitrary. The routing results of the three runs are saved. Then, to implement the attraction specification, a boolean flag is set true if there exists some victim that is attracted by the attacker only in the arbitrary run, and not in the normal and trivial runs. The assertion requires that this boolean flag is false. Therefore, if the assertion is violated, the violating run represents a succesful attraction attack. To implement the interception specification, a constraint that the attacker has a routing path to the real destination should be added.

Code 1.1. Node procedures

```
AS_Proc(){
  check incoming announcement and set chosen path;
  if(chosen path was changed)
    export new chosen path;
}
Arbitrary_Attacker_Proc(){
  Path p = nondeterministic_path();
  Neighbors G = nondeterministic_neighbors();
  foreach(n in G)
    send p to n;
}
Trivial_Attacker_Proc(){
  //attacker pretends to be dest
  Path p =<attacker>;
  send p to all neighbors;
}
Dest_Proc(){
  Path p =<dest>;
  send p to all neighbors ;
}
```

Code 1.2. BGP run

```
enum RunType {normal, trivial, arbitrary} ;
typedef Map<Node,Path> Routing_Results ;
Topology fragment;

Routing_Results BGP_run(RunType type){
  clear AS states;
  Dest_Proc();
  while(!stable_state())  {
    for(AS in fragment){
      if (AS is attacker and type == trivial)
        Trivial_Attacker_Proc();
      else if(AS is attacker and type == arbitrary)
        Arbitrary_Attacker_Proc();
      else
        AS_Proc();
    }
  }
  return routing results; //chosen paths of all nodes
}
```

Code 1.3. Main function with attraction specification

```
Routing_Results results[3];
int main(){
  results[normal] = BGP_run(normal);
  results[trivial] = BGP_run(trivial);
  results[arbitrary] = BGP_run(arbitrary);
  bool isSomeVictimAttracted = false;
  for(AS in fragment){
    if(AS routes via attacker in arbitrary run and not
          in normal and trivial runs)
      isSomeVictimAttracted = true;
  }
  assert(!isSomeVictimAttracted);
}
```

6.4 Applying model checking to the implemented model using ExpliSAT

Here we explain squares 5-7 of Figure 3. After the C code of the model is generated on the fragment, we apply model checking using ExpliSAT [7]. The model checker systematically scans all possible execution paths of the C program. If it finds a run that violates the assertion, it returns a counterexample that represents a successful attack. If the model checker terminates without any counterexample, it is considered a proof that our attacker cannot perform the specified attack on the fragment. This is denoted as *MC-proof.*

7 Experimental results

We applied our BGP-SA method on Internet fragments and used IBM's model checking tool ExpliSAT [7] to search for traffic attacks. The model checker can run on multiple cores. The experiments were performed on a 64-cores machine with AMD Opteron(tm) Processor 6376, 125GB RAM, and 64-bit Linux. The fragments and all model implementations we used in our experiments are available at [1].

ExpliSAT model checker ExpliSAT [7] verifies C programs containing assumptions and assertions. To use ExpliSAT we implement our model in C. Our specifications are negated and added as assertions on stable states. The model checker returns a counterexample if there is a violating run, and it can also perform *full verification* and automatically prove that no violating run is possible.

ExpliSAT combines explicit state model checking and SAT-based symbolic model checking. It traverses every feasible execution path of the program, and uses a SAT solver to verify assertions. It performs as many loop iterations as needed, and therefore full verification is possible and no loop bounds are required.

7.1 Results on Internet fragments

We performed experiments on self-contained fragments extracted from the full Internet topology. The ASes links from the Internet are from [5] and are relevant to October 2014.

Table 1 presents the results of applying our method. The fragments in lines 1-13 are based on randomly chosen destination and attacker from the Internet, with the exception of line 12 which is obtained by choosing the attacker and destination according to a recent attack where Syria attracted traffic destined to Youtube [18]. Line 14 is explained in Section 7.2. The first two columns specify the number of nodes in the extracted self-contained fragment and in the reduced fragment. The third column specifies the amount of nodes attracted by the attacker on the trivial attack. The value is invalid if the specification is interception and the trivial attack does not satisfy the interception condition, by which the attacker should have an available routing path to the destination. The specification we used for each instance appears on the fourth column, and is either attraction or

	Fragment size (#nodes)	Reduced size (#nodes)	Trivial attraction (#nodes)	Specification	Result	Time (min)	Dest ASN	Attacker ASN
1	16	11	9	attraction	BT proof	-	31132	16987
2	17	6	4	attraction	BT proof	-	9314	7772
3	22	10	8	attraction	BT proof	-	11669	36291
4	29	9	5	attraction	MC proof	1.5	29117	15137
5	15	13	10	attraction	MC proof	1	12431	18491
6	36	18	7	attraction	MC proof	17	19969	13537
7	69	27	17	attraction	MC proof	340	8296	20091
8	15	13	invalid	interception	counterexample	0.1	12431	18491
9	28	10	invalid	interception	counterexample	0.5	19361	32977
10	80	48	invalid	interception	counterexample	13	9218	43571
11	81	31	invalid	interception	counterexample	9	37177	40473
12	114	30	invalid	interception	counterexample	18	36040	29386
13	71	68	65	interception	N/A	>12h	30894	1290
14	10	-	4	interception	counterexample	0.1	-	-

Table 1. Results of BGP-SA application on fragments extracted from the full Internet topology

interception, which correspond to the specifications defined in Section 4.1. Note that in the interception specification, if the trivial attack fails to satisfy the interception condition, we only compare the attraction to the normal outcome. The result column specifies any of the possible results that are described in Section 6. The N/A result describes ExpiSAT runs that did not terminate. The last two columns specify the chosen ASN from the Internet of the destination and attacker nodes, from which the fragment was extracted.

The experiments show that the reductions we apply are significant. The simple BGP simulations of the trivial attack allow us to avoid applying model checking on fragments in which the attacker manages to achieve optimal attraction results by the trivial attack.

When we used ExpliSAT with the attraction specification, we got proofs that no better attack strategy exists. It can be explained by the fact that the trivial attack strategy can be considered most efficient in many cases. Consider for instance line 4 on which we got a proof by ExpliSAT. It should be noted that 2 nodes in the fragment are not trivially attracted and do not have definite routing choices, but still there is no attack strategy capable of attracting traffic from them. Thus, these two nodes are also considered *safe*, in addition to the nodes with definite routing choices.

For the interception instances in lines 8-12 the trivial attack failed to achieve the interception goal and ExpliSAT found simple interception attacks. Line 12 was performed on a fragment from a recent attack [18]. The fragment reduction was significant in this case. We found that the trivial attack attracted 12 nodes but did not satisfy the interception condition. The model checker found an attack strategy that achieved interception and attracted 11 nodes. The attacker sent false announcements to 3 of its 4 neighbors in the found interception attack.

7.2 Example demonstrating model checking advantages

Here we explain line 14 in the table. The network is taken from Figure 1. The network is a variation of the one presented in [9], where the goal was to show a non-trivial interception attack. We did not apply our reductions on this network topology.

In the normal outcome and trivial attack, the attacker fails to attract traffic from $AS8$. In the attack strategy suggested in [9] the attacker avoids exporting its path to $AS2$, and only exports it to $AS7$. The result is that $AS7$ chooses a shorter path directly via the attacker, and as a result $AS8$ prefers this shorter path. Thus, the attacker manages to apply traffic interception on $AS8$.

Line 14 of Table 1 specifies the experiment we performed on this topology with our BGP model. ExpliSAT automatically found a counterexample with greater attraction. It returned a counterexample in which the attacker exported announcements both to $AS7$ and to $AS2$. The announcement exported to $AS2$ contained $AS9$ on the sent path. Therefore, $AS9$ ignored that announcement, and did not export it to $AS7$. Thus, $AS7$ chose the shorter path via the attacker. Eventually, the attacker managed to achieve attraction from $AS8$, $AS2$, and $AS3$. Note that with the strategy suggested by [9] only $AS8$ is attracted. An alternative attack that could attract even more nodes to the attacker is to export to $AS2$ an announcement that contains $AS7$ instead of $AS9$ on the sent path. That way it can achieve attraction from $AS9$ as well.

From the above analysis we may conclude that by sending an announcement that creates a loop an attacker can better control on where the propagation of some path should be blocked in order to achieve better attraction results.

It should be noted that some versions of BGP are more secure [10] and may prevent the attacker from sending paths that do not exist in the network. On such versions the attacker cannot apply the loop strategy. Therefore, the loop strategy may have an advantage over the no-export strategy only in the absence of certain BGP security mechanisms.

8 Conclusion

In this work we propose a method to reveal possible attacks on Internet routing or prove that certain attacks are not possible. We develop substantial reduction techniques that enable to apply model checking in order to formally analyze BGP traffic attacks on the Internet. The use of model checking has a major advantage due to the systematic search, by which it can reveal unexpected or more sophisticated attacks. This is demonstrated in Section 7.2, where during an experiment that was done to reconstruct a known attack, the model checker automatically found a different attack strategy that achieved better attraction results than expected.

One obvious implication of our work is a better understanding of the vulnerability of the Internet to traffic attacks. Nonetheless, our suggested method can also be practical and useful for a network operator to increase its resilience to such attacks. In some cases a network operator may fear a traffic attack from potential attacking ASes. For example, telecommunication companies may fear their traffic be attracted by ASes that belong to adversary governments. Such governments can exploit these attacks in order to eavesdrop on traffic of consumers of those telecommunication companies. In such cases, the network operator can use our method in order to discover the identity of the ASes which the attacking AS can not attract traffic from. Once these *safe* ASes are known the network operator may form links to these ASes and prefer routes announced by those ASes, thereby eliminate the chances to be attracted by the attacker.

Acknowledgement

The research was supported by The Prof. A. Pazy Research Foundation.

References

1. http://www.cs.technion.ac.il/˜sadisos/bgp/.
2. Matvey Arye, Rob Harrison, and Richard Wang. The next 10,000 bgp gadgets.
3. Matvey Arye, Rob Harrison, Richard Wang, Pamela Zave, and Jennifer Rexford. Toward a lightweight model of bgp safety. *Proc. of WRiPE*, 2011.
4. Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 265–276. ACM, 2007.
5. CAIDA. Inferred AS Relationships Dataset. http://data.caida.org/datasets/as-relationships/serial-1/20141001.as-rel.txt.bz2, October 2014.
6. R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. IETF RFC 1195, December 1990.
7. Hana Chockler, Dmitry Pidan, and Sitvanit Ruah. Improving representative computation in ExpliSAT. In *Haifa Verification Conference (HVC)*, LNCS 8244, Haifa, Israel, 2013.
8. Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking (TON)*, 9(6):681–692, 2001.
9. Sharon Goldberg, Michael Schapira, Peter Hummon, and Jennifer Rexford. How secure are secure interdomain routing protocols? *Computer Networks*, 70:260–287, 2014.
10. Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure border gateway protocol (s-bgp. *IEEE Journal on Selected Areas in Communications*, 18:103–116, 2000.
11. Robert Lychev, Sharon Goldberg, and Michael Schapira. Network-destabilizing attacks. *arXiv preprint arXiv:1203.1681*, 2012.
12. Doug Madory. Sprint, Windstream: Latest ISPs to hijack foreign networks. http://research.dyn.com/2014/09/latest-isps-to-hijack/, September 2014.
13. Doug Madory. The Vast World of Fraudulent Routing. http://research.dyn.com/2015/01/vast-world-of-fraudulent-routing/, January 2015.
14. G. Malkin. RIP version 2. IETF RFC 2453, November 1998.
15. J. Moy. OSPF version 2. IETF RFC 2328, April 1998.
16. Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). IETF RFC 4271, January 2006.
17. Yiqing Ren, Wenchao Zhou, Anduo Wang, Limin Jia, Alexander JT Gurney, Boon Thau Loo, and Jennifer Rexford. Fsr: formal analysis and implementation toolkit for safe inter-domain routing. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 440–441. ACM, 2011.
18. Andree Toonk. BGP hijack incident by Syrian Telecommunications Establishment. http://www.bgpmon.net/bgp-hijack-incident-by-syrian-telecommunications-establishment/, December 2014.
19. Andree Toonk. Hijack event today by Indosat. http://www.bgpmon.net/hijack-event-today-by-indosat/, April 2014.
20. Andree Toonk. The Canadian Bitcoin Hijack. http://www.bgpmon.net/the-canadian-bitcoin-hijack/, August 2014.
21. Pierre-Antoine Vervier, Olivier Thonnard, and Marc Dacier. Mind your blocks: On the stealthiness of malicious BGP hijacks. 2015.