UNIVERSITY OF CALIFORNIA, SAN DIEGO

Investigating an Approach for Discovering Second Hop Neighbors in BGP

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Jae Hyun Park

Committee in charge:

Kimberly C. Claffy, Chair
Alex C. Snoeren, Co-Chair
Alberto Dainotti
Aaron D. Shalev

2017

The Thesis of Jae Hyun Park is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Alberto Dainotti, for all the opportunities and guidance throughout the Master's program. I am honored of being in part of CAIDA.

I would also like to thank my committee members, kc claffy, Alex Snoeren, and Aaron Schulman for reviewing my thesis.

Finally, I would like to thank my family and friends for unconditional love and support.

Chapter 1, in part, is currently being prepared for submission for publication of the material. Sermpezis, Pavlos; Kotronis, Vasileios; Gigis, Petros; Dimitropoulos, Xenofontas; Park, Jae Hyun; Cicalese, Danilo; King, Alistair; Dainotti, Alberto. The thesis author is a co-author of this paper [SKG$^+$].

Chapter 2, in part, is currently being prepared for submission for publication of the material. Sermpezis, Pavlos; Kotronis, Vasileios; Gigis, Petros; Dimitropoulos, Xenofontas; Park, Jae Hyun; Cicalese, Danilo; King, Alistair; Dainotti, Alberto. The thesis author is a co-author of this paper [SKG$^+$].

Chapter 3, in part, is currently being prepared for submission for publication of the material. Sermpezis, Pavlos; Kotronis, Vasileios; Gigis, Petros; Dimitropoulos, Xenofontas; Park, Jae Hyun; Cicalese, Danilo; King, Alistair; Dainotti, Alberto. The thesis author is a co-author of this paper [SKG$^+$].

ABSTRACT OF THE THESIS

Investigating an Approach for Discovering Second Hop Neighbors in BGP

by

Jae Hyun Park

Master of Science in Computer Science

University of California, San Diego, 2017

Kimberly C. Claffy, Chair
Alex C. Snoeren, Co-Chair

BGP prefix hijacking, which is illegitimate takeover of IP prefixes by announcing forged AS paths, is a major threat to the Internet. A number of hijacking events with severe consequences in the Internet routing system have been documented. Several studies have proposed techniques for network operators to detect hijacking autonomously to quickly react to attacks. In this thesis, we focus on autonomously detecting one of the impactful types of hijacks, a hijack event in which a forged AS is placed two AS-hops from the origin AS in AS paths. Thus, we develop an approach to discover second hop neighbors of an AS owner based on carefully crafted BGP announcements, and use this information as a baseline to evaluate anomalies in AS paths and detect hijacking events. *Second hop neighbors* of an AS are ASes two hops from the origin AS in a BGP-observed topology. An AS owner can quickly classify an announcement as legitimate if the ASN that is two AS-hops from the origin ASN in an AS path is in its set of second hop neighbors. Thus, the more second hop neighbors an AS owner discovers, the more AS paths it can correctly classify as legitimate announcement, resulting in less false-positive rate. Through simulation experiments, we show that our approach finds more than 80% of second hop neighbors for 80% of origin ASes that we study.

Chapter 1

Introduction

This thesis investigates an approach for discovering second hop neighbors of an autonomous system (AS) in Border Gateway Protocol (BGP). BGP is the de-facto Internet inter-domain routing protocol, controlling routing paths between interconnected ASes. As of May 2017, there are more than 57,000 unique ASes [BSH17], forming a complex AS topology. *Second hop neighbors* of an AS are ASes two hops from the origin AS in a BGP-observed AS topology. By extending this notion, we define *n-hop neighbors* of an AS, which are ASes that are n AS-hops from the origin AS in a BGP-observed topology, where $n \geq 1$.

BGP prefix hijacking is illegitimate takeover of an IP prefix due to a malicious or misconfigured router either originating a prefix that it does not own (origin-AS hijack), or announcing an illegitimate path for such a prefix (false link hijack). Due to security gaps in the design of BGP, which allows any AS to announce illegitimate routes in the Internet, BGP hijacking is a major threat to Internet operators and users. Several BGP hijacking events have been documented [CGHS16, GCHS17], some leading to severe consequences with the Internet routing system and economy. Fast detection of BGP hijacking is essential for preventing the attack from resulting in severe consequences.

Currently, many network operators use third-party services to detect hijacking events [SKG$^+$]. These services notify the operators upon hijacking events detection, which does not allow fast mitigation.

The main motivation for discovering second hop neighbors is for the AS owner to autonomously and quickly detect BGP prefix hijacking attacks against it. From the perspective of an AS, it is possible to detect a hijacking attack of its own prefix by (i) knowing its n-hop neighbors and (ii) scanning for suspicious BGP announcements using route monitors. Route monitors are operational routers that peer with route collectors, which are hardware or software routers that collect and store BGP route announcements [oO17, NCC17]. These monitors provide a global view of BGP routes announced on the Internet. Ideally, an AS owner can use real-time monitors and knowledge of its n-hop neighbors to detect hijacking events seen by at least one monitor, using this knowledge as a baseline to evaluate anomalies in AS paths and to classify BGP announcements as suspicious or legitimate.

The BGP Monitoring Protocol (BMP) [JSNF$^+$16] allows BMP-enabled route collectors to receive all BGP routes from its BGP peers in real-time using BMP messages. Currently, however, BMP-enabled monitors placed around the world are in their experimental stage [CAI17]. Thus, in this thesis, we use near real-time route monitors, such as Route Views [oO17] and RIPE Routing Information Service (RIS) [NCC17].

We focus on detecting impactful types of BGP hijacking events. We refer to origin-AS hijacking as Type-0 hijacking, and false link hijacking as Type-N ($N \geq 1$), where N is the number of hops from the hijacker's AS number (ASN) to the origin ASN in the announced path. Also, we define an AS as *polluted* if it selects a path that contains the ASN of the hijacker according to its routing policy. [SKG$^+$] quantifies the impact of a hijacking event by the percentage of polluted ASes due to the event, and shows that Type-0, Type-1, and Type-2 hijacks have large impact on the Internet. These types of hijacks tend to be more impactful because the attacker generates a shorter path compared to a Type-N ($N \geq 3$) hijacks; therefore, there is a higher chance that other routers will prefer the forged path over the legitimate path, thus, letting it propagate further. An AS owner can easily detect Type-0 and Type-1 hijacks because the owner knows not only which ASes are allowed to originate its prefixes, but also its first hop neighbors. However, the owner cannot easily detect Type-2 hijacks without knowledge of its second hop neighbors. Therefore, in this thesis, we focus on the development and evaluation of an approach for discovering second hop neighbors.

The simplest way to discover second hop neighbors is to scan all route monitors and extract second hop neighbors from the AS paths. However, using this approach, we cannot discover *hidden second hop neighbors*, which are connected through back-up paths or less-preferred paths. If we use this set of second hop neighbors as a baseline to evaluate anomalies in AS paths, we will suffer from a high false-positive rate because routers may include hidden second hop neighbors in AS paths due to legitimate routing changes, such as policy changes or outages.

Thus, we have devised an approach using BGP poisoning [Col06]. *BGP poisoning* is a technique that involves announcing carefully crafted BGP messages to discover alternative routes that do not traverse certain ASes. ASes that should not be traversed (*poisoned*) are prepended in a BGP announcement of a prefix p so that the AS path is rejected when the message reaches such ASes due to BGP's loop prevention mechanism. These ASes then withdraw their paths from their neighbors since they do not have a valid route towards prefix p. We use this technique to reveal alternative routes and discover hidden second hop neighbors. We evaluate our approach in terms of (i) *discovery coverage*, percentage of second hop neighbors discovered, and (ii) *deployment feasibility*, the maximum required length of AS paths in poisoned BGP announcements.

The contributions in this thesis are as follows:

- Developing various approaches to discover second hop neighbors in BGP.
- Evaluating these approaches in terms of discovery coverage using simulation, and deployment feasibility using real world data.

The remaining chapters in this thesis are structured as follows. We discuss background and related work in Chapter 2. In Chapter 3, we explain data and tools used in this thesis. In Chapter 4, we propose and discuss various approaches to discover second hop neighbors in BGP. In Chapter 5, we perform simulation experiments of the approaches discussed in Chapter 4 and discuss our results. In Chapter 6, we discuss benefits and limitations of these approaches and future work. Lastly, we conclude the thesis in Chapter 7.

1.1 Acknowledgement

Chapter 1, in part, is currently being prepared for submission for publication of the material. Sermpezis, Pavlos; Kotronis, Vasileios; Gigis, Petros; Dimitropoulos, Xenofontas; Park, Jae Hyun; Cicalese, Danilo; King, Alistair; Dainotti, Alberto. The thesis author is a co-author of this paper [SKG+].

Chapter 2

Background and Related Work

In this chapter, we review BGP prefix hijacking and BGP poisoning. Then, we discuss related work.

2.1 BGP Prefix Hijacking

BGP prefix hijacking is illegitimate takeover of IP prefixes using bogus advertisement in BGP. It can occur due to a malicious or misconfigured router originating a prefix, or announcing an illegitimate path for a prefix not owned by the AS operating the router. Due to security gaps in the design of BGP, which allows any ASes to announce illegitimate routes in the Internet, routers may propagate the illegitimate announcement and route traffic to the hijacker. After receiving the traffic, the hijacker may eavesdrop, steal, or modify the traffic. BGP prefix hijacking is also referred to as *BGP hijacking*, *prefix hijacking* or *IP hijacking*.

We review types of BGP hijacking events that we use as reference in the rest of this thesis. For the sake of illustration, we assume the following: (i) AS *O* owns and legitimately announces the prefix *10.0.0.0/23*, and (ii) AS *H* is the hijacker's AS number. We denote a BGP message with two fields: its AS path and announced prefix.

For example, [X Y O | *10.0.0.0/23*] is a BGP announcement message for the prefix *10.0.0.0/23* with AS path [X Y O] originated by the legitimate AS O. Also, we define an AS as *polluted* if it selects a path that contains the ASN of the hijacker according to its routing policy.

2.1.1 Hijacking Types: Announced Path

Origin-AS Hijacking

In origin-AS hijacking, a hijacker announces a prefix that it is not authorized to originate. In our example, neighboring ASes of AS H would receive an illegitimate BGP announcement, [H | 10.0.0.0/23], and propagate this announcement. Then, these polluted ASes would route their traffic to the hijacker. Origin-AS hijacking is the most commonly observed hijacking type, and may occur due to an attack or a misconfiguration, which we also call Type-0 hijacking.

False Link Hijacking

In false link hijacking, a hijacker deliberately originates an illegitimate path for a prefix it does not own. The path announced by the hijacker contains both its AS number as the last hop and the sequence of ASes that constitutes an illegitimate AS path. An example of fake link hijacking announced by AS H is [H X Y O | *10.0.0.0/23*], where AS H is not a neighbor of AS X. As a result, the polluted ASes route their traffic through the hijacker, which can then eavesdrop, steal, or modify the traffic.

We refer to a false link hijacking event as Type-N ($N \geq 1$), where N is the number of hops from the hijacker's AS number (ASN) to the origin AS number in the announced illegitimate path.

2.1.2 Hijacking Types: Affected Prefix

Exact Prefix Hijacking

In *exact prefix hijacking*, a hijacker originates a path for the exact same prefix that the legitimate AS announces. In this case, only *a portion of ASes in the Internet* that prefer routes to the hijacker are polluted, switching the routes towards the hijacker. Typically, the ASes that are closer to the hijacker in number of AS-hops are polluted, since BGP route selection algorithms executed at each router prefer shorter paths.

Sub-prefix Hijacking

In *sub-prefix hijacking*, a hijacker announces a more specific prefix that is covered by the prefix of the legitimate AS. For example, the hijacker announces a path [*H* | *10.0.0.0/24*] or [*H X Y O* | *10.0.0.0/24*]. Since more specific prefixes are preferred in BGP routing, *all ASes in the Internet* are polluted, switching routes toward the hijacker for the announced sub-prefix.

2.2 BGP Poisoning

BGP poisoning is a technique to find alternative BGP routes that do not traverse certain ASes by announcing carefully crafted BGP messages. It leverages the AS loop prevention mechanism in BGP [RLH06]. An AS prepends ASes that should not be traversed (*poisoned*) in a BGP announcement. These ASes, upon receiving the crafted announcement, then reject the AS path and withdraw their paths from their neighbors due to BGP's loop prevention mechanism. For example, to find routes that do not traverse AS *P* towards a prefix *p* owned by AS *O*, AS *O* can announce an AS path that includes *P*, such as [*O P O* | *p*]. When this announcement reaches AS *P*, AS *P* rejects the path and withdraws its path from neighbors because propagating this announcement by prepending

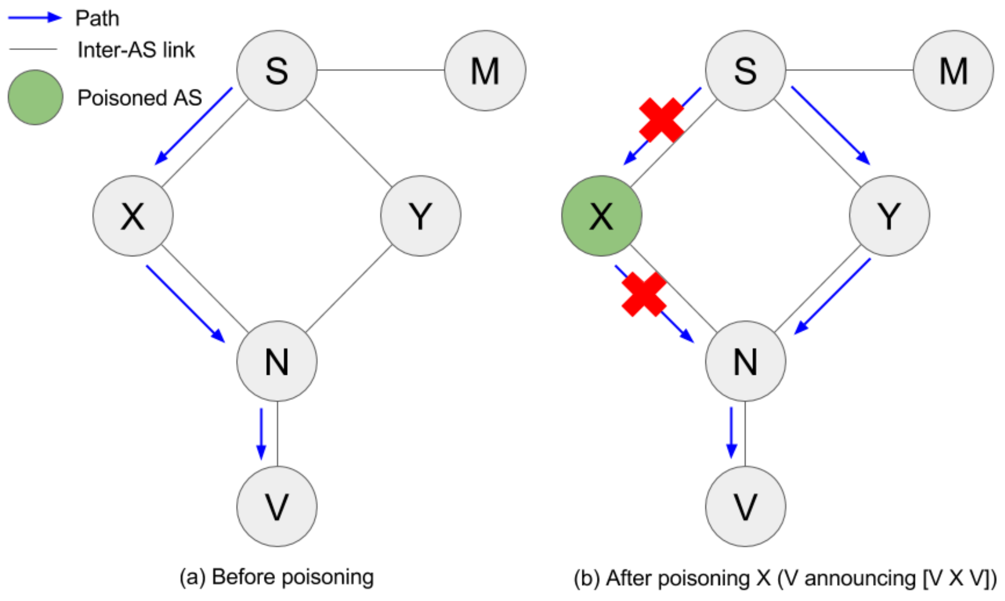


Figure 2.1: An example of BGP poisoning technique. AS V announces [V X V] to poison AS X.

its ASN results in a loop in the AS path. Thus, no routes towards the prefix p will traverse AS P: AS P is *poisoned*.

We can also poison multiple ASes at the same time. One simple approach is to include all ASes we intend to poison in the AS path of the announcement, such as [O P_1 P_2 ... P_n O], where P_1, P_2, ... P_n are the ASes to poison. However, we cannot use this approach when the number of ASes to poison is large because BGP routers filter out BGP messages with long AS paths. Another approach is to use an AS set, which is an unordered set of AS numbers, to include all ASes to poison in the AS path, such as [O $\{P_1, P_2, \ldots, P_n\}$ O], where $\{P_1, P_2, \ldots, P_n\}$ is an AS set with ASes to poison. By using this approach, which is known as *AS-set stuffing* [Col06], we always send BGP announcements with lengths of 3 to poison any number of ASes. However, this approach is also problematic because BGP routers may filter out messages with a large AS set in an AS path. To maximize the number of ASes to poison, we need to use multiple AS sets in AS paths. We can calculate the maximum number of ASes to poison in a BGP announcement as follows:

$$P_{max} = (L_{max} - 2) \times S_{max}, \tag{2.1}$$

where P_{max} is the maximum number of ASes we can poison, L_{max} is the maximum length of the AS path in a BGP announcement, and S_{max} is the maximum size of an AS set in a BGP announcement. Currently, there are no standardized limits for the length of the AS path and the size of AS sets in a BGP announcement. In this thesis, we use 6 as the limit for both the length of the AS path and the size of AS sets, allowing us to poison up to 24 ASes by announcing [O $\{P_1$, ... ,$P_6\}$ $\{P_7$, ..., $P_{12}\}$ ... $\{P_{19}$, ..., $P_{24}\}$ O].

An example of BGP poisoning is illustrated in Figure 2.1. In this simple topology, the monitor M observes a path [$S\ X\ N\ V \mid p$] towards a prefix p announced by AS V. To find alternative routes from AS S toward p, we poison AS X by announcing [$V\ X\ V \mid p$]. The loop prevention mechanism withdraws the paths from AS X to AS V, causing AS S to choose an alternative route to the prefix p, which is [$S\ Y\ N\ V \mid p$].

BGP poisoning allows us to discover alternative routes towards a prefix, which will help us to discover second hop neighbors of an AS. From the newly discovered routes, we may find new second hop neighbors of an AS. We use BGP poisoning as our main approach to discover second hop neighbors.

2.3 Related Work

A number of systems have been proposed for detecting BGP prefix hijacking based on control plane [LMP+06, QGRN07] or data plane [ZJP+07, ZZH+08] or both [HM07]. However, most of them are designed to operate as third-party services that monitor the Internet and notify involved ASes upon detection of suspicious incidents. In addition, hijacking detection techniques based on control plane BGP data do not use *n*-hop neighbors as baselines to evaluate anomalies in AS paths. A number of studies use BGP poisoning (Section 2.2) to find alternative BGP paths [Col06, ANC$^+$15, JCC$^+$13, KBSC$^+$12]. However, none of these studies focus on detecting BGP hijacking events.

This thesis work is the first step to explore how an AS owner can detect BGP hijacking events autonomously by leveraging the knowledge of second hop neighbors discovered using BGP poisoning.

2.4 Acknowledgement

Chapter 2, in part, is currently being prepared for submission for publication of the material. Sermpezis, Pavlos; Kotronis, Vasileios; Gigis, Petros; Dimitropoulos, Xenofontas; Park, Jae Hyun; Cicalese, Danilo; King, Alistair; Dainotti, Alberto. The thesis author is a co-author of this paper [SKG$^+$].

Chapter 3

Data and Tools

In this chapter, we discuss about data and tools we use throughout the discovery of second hop neighbors.

3.1 CAIDA AS Relationship Data

CAIDA AS relationship data [LHD$^+$13] provides a set of inferred relationships between ASes. The AS relationships are inferred from BGP paths in the routing table snapshots collected by the Route Views [oO17] and RIPE RIS [NCC17] monitors. The dataset contains a list of AS pairs with a peering link, which is annotated based on their relationship as provider-to-customer or peer-to-peer.

We use CAIDA AS relationship data in the BGP simulator (Section 3.5). AS relationship data is necessary to simulate BGP routing policies, such as the Gao-Rexford model [GR01].

3.2 Data Concierge

Data Concierge is a Python library that allows effective and efficient use of CAIDA AS relationship data. At initialization, it reads CAIDA AS relationship data, parses them, and stores them in an efficient manner to retrieve data. Some useful interfaces include: get_customer_cone(), in_customer_cone(), get_relationships(), get_common_neighbors(). In this work, we use the in_customer_cone() and get_relationships() interfaces.

3.3 Route Views and RIPE RIS

The Route Views project [oO17], developed by University of Oregon, provides near real-time information about the global routing system through Route Views routers that peer with Internet transit providers. Route Views route collectors collect a full BGP routing information base (RIB) every 2 hours and BGP updates every 15 minutes. All collected data are archived in the multi-threaded routing toolkit routing information export format (MRT format) [BKN+11].

The RIPE's Routing Information Service (RIS) [NCC17] also provides Internet routing data collected from routers in several locations around the globe. The data can be accessed via the raw files in MRT format. RIPE RIS route collectors collect full BGP RIB every 8 hours and BPG updates every 5 minutes.

Throughout this work, we use AS paths provided by the Route Views and RIPE RIS monitors that are placed in 258 different ASes around the world.

3.4 BGPStream

BGPStream [OKG+16] is an open-source software framework, providing an easy and efficient way of processing large amounts of live and historical BGP measurement data. It provides a set of APIs and tools to process large amounts of BGP measurement data, which are often used in scientific research, operational monitoring, and post-event analysis. There are several sources of BGP measurement data, including Route Views and RIPE RIS collectors. This framework provides two programming APIs for accessing BGP measurement data: libBGPStream, a C API, and PyBGPStream, a Python API.

Throughout this work, we use PyBGPStream to read RIB dumps and updates from Route Views and RIPE RIS route collectors.

3.5 BGP Simulator

The BGP simulator [SKG$^+$], developed by the Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH), considers ASes as single nodes, and simulates BGP routing among them. Specifically, it simulates (i) BGP message exchanges between nodes and (ii) selection of BGP paths from nodes based on routing policies. The BGP simulator uses CAIDA AS relationship data (Section 3.1) to infer AS relationships, provider-to-customer or peer-to-peer, between nodes. For the routing policies between ASes, it simulates the Gao-Rexford model [GR01]. The simulator is implemented in Python3. We use this simulator in all our simulation experiments to discover second hop neighbors.

3.6 PEERING Testbed

PEERING [PEE17] is a BGP testbed for researchers to conduct Internet routing experiments, such as announcing/selecting routes and sending/receiving traffic. It is connected with real networks via BGP at universities and Internet exchange points (IXPs) around the world, including Amsterdam Internet Exchange, Phoenix Internet Exchange, Georgie Institute of Technology, University of Washington, and Universidade Federal de Minas Gerais.

To use the PEERING testbed, researchers must submit proposals describing their experiment. Upon project approval, they are given a set of prefixes to experiment on. The PEERING testbed administrators restrict the frequency of BGP announcement to one BPG announcement modification per 90 minutes to prevent experiment from disrupting the Internet. For a similar reason, the length of BGP announcement is restricted to ≤ 5.

The PEERING testbed currently uses the BIRD Internet Routing Daemon (BIRD) to control BGP route announcements and withdrawals. Although this is the default tool provided by the the PEERING testbed administrators, we can send announcements through other BGP implementations, such as ExaBGP [EN17] and Quagga [Qua17].

3.7 Acknowledgement

Chapter 3, in part, is currently being prepared for submission for publication of the material. Sermpezis, Pavlos; Kotronis, Vasileios; Gigis, Petros; Dimitropoulos, Xenofontas; Park, Jae Hyun; Cicalese, Danilo; King, Alistair; Dainotti, Alberto. The thesis author is a co-author of this paper [SKG$^+$].

Chapter 4

Approaches to Discover Second Hop Neighbors

We discover second hop neighbors of an AS by using near real-time route monitors, such as Route Views [oO17] and RIPE RIS [NCC17] monitors. The simplest way to discover second hop neighbors of an AS is to scan all AS paths from the route monitors. However, using this approach, we only discover second hop neighbors that are connected through active, or “best”, paths. To discover hidden second hop neighbors, we propose to use BGP poisoning (Section 2.2), which is a technique to reveal alternative BGP routes using carefully crafted BGP messages. With BGP poisoning, we can discover a large set of second hop neighbors of an AS.

An AS owner needs knowledge of its second hop neighbors to autonomously detect BGP hijacking events of Type-2 (Section 2.1). Ideally, with knowledge of all second hop neighbors, an AS owner does not misclassify observation of a second hop neighbor that was previously hidden as a BGP hijacking event, since the owner is already aware that this AS is its second hop neighbor. The owner classifies observation of a new second hop neighbor that is not in its set of second hop neighbors as a suspicious event.

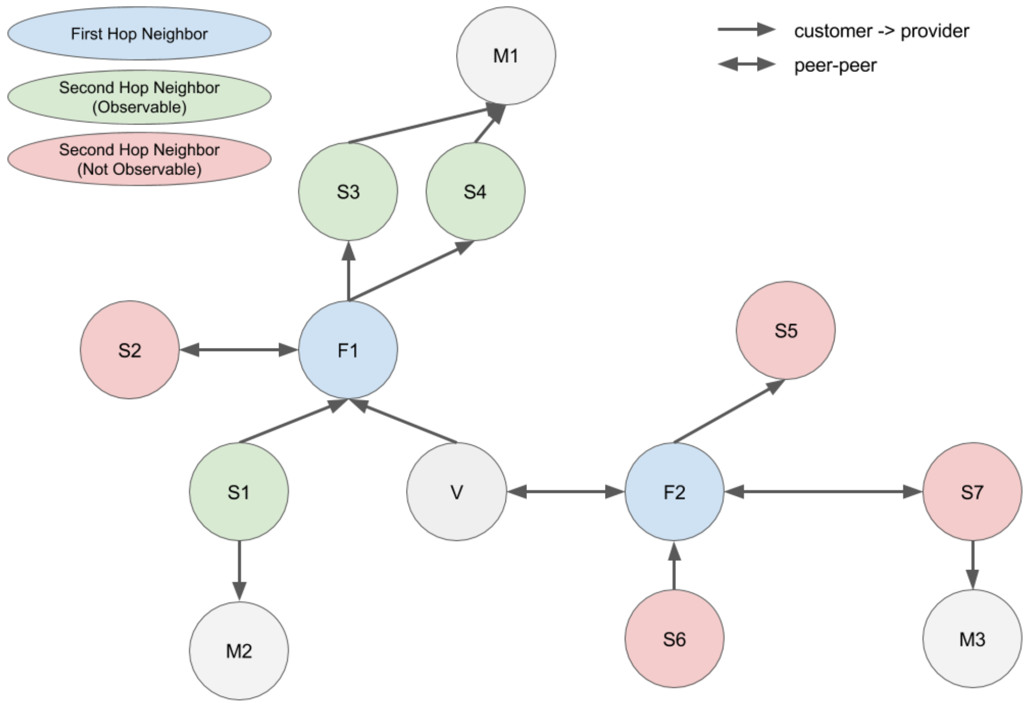


Figure 4.1: An example of AS topology illustrating observable second hop neighbors of an AS. V is the origin AS, $F1$ and $F2$ are first hop neighbors of V, $S1-S7$ are second hop neighbors of V, and $M1-M3$ are monitors.

This suspicious event may be due to an illegitimate announcement or to the establishment of a new link or BGP peering session. Distinguishing whether new second hop neighbors are observed due to illegitimate announcements or to the establishment of new links or BGP peering sessions is out of the scope of this thesis.

4.1 Observable and Observed Second Hop Neighbors

An AS owner cannot observe all its second hop neighbors through the route monitors. The observability of the second hop neighbors depends on the route export policy and placement of the monitors in the AS topology. We first define *observable*, *non-observable*, and *observed* second hop neighbors.

- *Observable* second hop neighbor: a second hop neighbor that can be observed from at least one monitor at some point in time. Usually, the monitor is in the provider chain or in the customer cone of an *observable* second hop neighbor.
- *Non-observable* second hop neighbor: a second hop neighbor that cannot be observed from monitors at any time due to monitor placement and existing AS relationships, if all BGP announcements are legitimate.
- *Observed* second hop neighbor: a second hop neighbor that is observed from monitors in a specific simulation / emulation experiment.

Figure 4.1 illustrates examples of observable and non-observable second hop neighbors. In this example, AS V is the prefix owner, AS $F1$ and AS $F2$ are first hop neighbors of AS V, AS $S1$-$S7$ are second hop neighbors of AS V, and AS $M1$-$M3$ are the monitors. Assume a prefix p is announced by AS V, and BGP routing follows the Gao-Rexford model [GR01]. In the remainder of this section, we only use the AS label for brevity. $F1$ is a provider of V, so it exports the received route to all of its neighbors. Similarly, $S3$ and $S4$ export the received route to all of their neighbors because $S3$ and $S4$ are providers of $F1$. Thus, the monitor $M1$, which is neighbor of both $S3$ and $S4$, receives the AS paths [$S3$ $F1$ V | p] and [$S4$ $F1$ V | p] from $S3$ and $S4$, respectively, and chooses preferred path according to its routing policy. Let us assume that $M1$ prefers [$S3$ $F1$ V | p]. As a result, the prefix owner sees the active AS path, [$M1$ $S3$ $F1$ V | p] from $M1$ monitor, and discovers that $S3$ is a second hop neighbor of V. In this case, $S4$ is also an observable second hop neighbor of V because it is a hidden second hop neighbor. At any point in time, $M1$ monitor may observe [$S4$ $F1$ V | p] due to policy changes in ASes, outages, link failures, or poisoning $S3$. In case of $S2$ and $S1$, they export routes only to their customers because they received routes from $F1$, which is their peer and provider, respectively. In this case, the prefix owner cannot observe the AS path from $S2$

to V through the monitors because there are no monitors in the customer cone [LHD+13] of $S2$. However, in case of $S1$, since there is a monitor $M2$ in the customer cone of $S1$, the prefix owner observes the AS path from $S1$ to V through $M2$, such as [$M2$ $S1$ $F1$ V | p], and discovers $S1$ as its second hop neighbor. If a first hop neighbor of V is a peer or a customer, such as $F2$, it exports the received route only to its customers. Thus, in the example, the routes from $S5$ to V and $S7$ to V towards prefix p do not exist, as $S5$ is a provider of $F2$ and $S7$ is a peer of $F2$. Therefore, the prefix owner cannot observe $S5$ and $S7$ through the monitors regardless of monitor placement. For a customer of $F2$, such as $S6$, the observability depends on the existence of monitors in the customer cone of the AS. Since there are no monitors in the customer cone of $S6$, the route from $S6$ to V is not observable through the monitors.

For the detection of Type-2 BGP hijacking events, we need to focus on discovering observable second hop neighbors. Let us assume that an AS owner discovered a complete set of its observable second hop neighbors. Then, in case of legitimate routing changes, which may be due to outages or policy changes, the owner will observe second hop neighbors that are already in its set. Thus, legitimate routing changes are not misclassified as suspicious events. When the owner sees a new, non-observable, second hop neighbor in a BGP announcement, the owner will classify it as a suspicious event. This suspicious event may be due to illegitimate announcements or to the establishment of new links and peering sessions. Thus, the owner can correctly classify illegitimate announcements and new links as suspicious events only with a set of observable second hop neighbors. Therefore, we focus only on exhaustively discovering observable second hop neighbors to detect these suspicious events.

In the next chapter, we proceed by performing simulation experiments of our approaches, where we evaluate the ability of different approaches to discover second hop neighbors. To correctly evaluate such a capability, we first need to know the set of

Algorithm 1: Identifying all observable second hop neighbors of an AS given full knowledge of the AS topology.

```
Data: prefix owner AS $V$, a set $S_{observable}$ (observable second hop neighbors
      of $V$), a set of monitor ASes $M$
1  $F_p$ = Providers of $V$
2  $F_{cp}$ = Customers and Peers of $V$
3  foreach $AS\ F\ in\ F_p$ do
4      $S_p$ = Providers of $F$
5      $S_{cp}$ = Customers and Peers of $F$
6      $S_{observable} = S_{observable} \cup S_p$
7      foreach $AS\ S\ in\ S_{cp}$ do
8          if Any $M$ in customer cone of $S$ then
9              Add $S$ to $S_{observable}$
10         end
11     end
12 end
13 foreach $AS\ F\ in\ F_{cp}$ do
14     $S_c$ = Customers of $F$
15     foreach $AS\ S\ in\ S_c$ do
16         if Any $M$ in customer cone of $S$ then
17             Add $S$ to $S_{observable}$
18         end
19     end
20 end
21 return $S_{observable}$
```

observable second hop neighbors in order to compare it with the set of observed second hop neighbors. Thus, we have devised an algorithm to identify all observable second hop neighbors of an AS given full knowledge of the AS topology (Algorithm 1). This algorithm classifies a second hop neighbor as observable if at least one monitor exists in the provider chain or in the customer cone of it. We assume that BGP routing policy follows the Gao-Rexford model [GR01]. Ideally, using BGP poisoning (Section 2.2), we would have equivalent sets of observable and observed second hop neighbors of an AS.

4.2 Approaches for Discovering Second Hop Neighbors

In this section, we present a series of algorithms to discover second hop neighbors of an AS. In all algorithms, we use the BGP poisoning approach (Section 2.2) in an attempt to discover all observable second hop neighbors. We also discuss limitations of each algorithm in terms of (i) second hop neighbor discovery coverage, and (ii) deployment feasibility. For deployment feasibility, we focus on the required length of AS path to perform our approaches because BGP announcements with long AS path may be filtered by routers. We present four approaches, each approach improving on either discovery coverage or deployment feasibility of the previously-introduced approach: (i) *naive poisoning approach*, which is not deployable and has small discovery coverage, (ii) *depth-first poisoning approach*, which is not deployable but has large discovery coverage, (iii) *poisoning first hops only approach*, which is deployable but has medium discovery coverage, and (iv) *poison by divide and conquer approach*, which is deployable and has large discovery coverage.

Algorithm 2: Discovering second hop neighbors of an AS by naively poisoning all second hop neighbors seen by the monitors.

```
    Data: Prefix Owner AS $V$, Prefix $P$ (announced by $V$), a set $S_observed$
          (observed second hop neighbors of $V$), a set of monitor ASes $M$
1   while Any $M$ has path to $P$ do
2       foreach monitor in $M$ do
3           for AS path to $P$ do
4               Add second hop neighbors of $V$ to $S_{observed}$
5           end
6       end
7       Announce AS path [$V$ $S_{observed}$ $V$] from $V$ to poison ASes in $S_{observed}$
8   end
9   return $S_{observed}$
```

4.2.1 Naive Poisoning Approach (Approach 1 - Naive)

One simple approach to discover second hop neighbors of a prefix owner AS, AS V, is to repeat the process of collecting all the second hop neighbors of AS V seen from the AS paths in the monitors and poisoning all the collected second hop neighbors until all monitors do not see any path towards the prefix originated from AS V, as described in Algorithm 2. We refer to this approach as the *naive poisoning approach* or *Approach 1 - Naive*.

Although this approach is simple and easy to implement, it fails to discover all observable second hop neighbors of a prefix owner AS. Figure 4.2 illustrates a limitation of *Approach 1 - Naive* in discovering second hop neighbors. In this topology, we observe paths [X B A V], [Y B A V], and [Z B A V] from the monitors, $M1$, $M2$, and $M3$, respectively. Then we poison AS B, which is discovered as a second hop neighbor of AS V. After AS B is poisoned, we cannot discover additional second hop neighbors. However, in the example topology, AS X, AS Y, and AS Z are also observable second hop neighbors of AS V. Therefore, this approach fails to discover AS X, AS Y, and AS Z as observed second hop neighbors of AS V. This shows that *Approach 1 - Naive* cannot

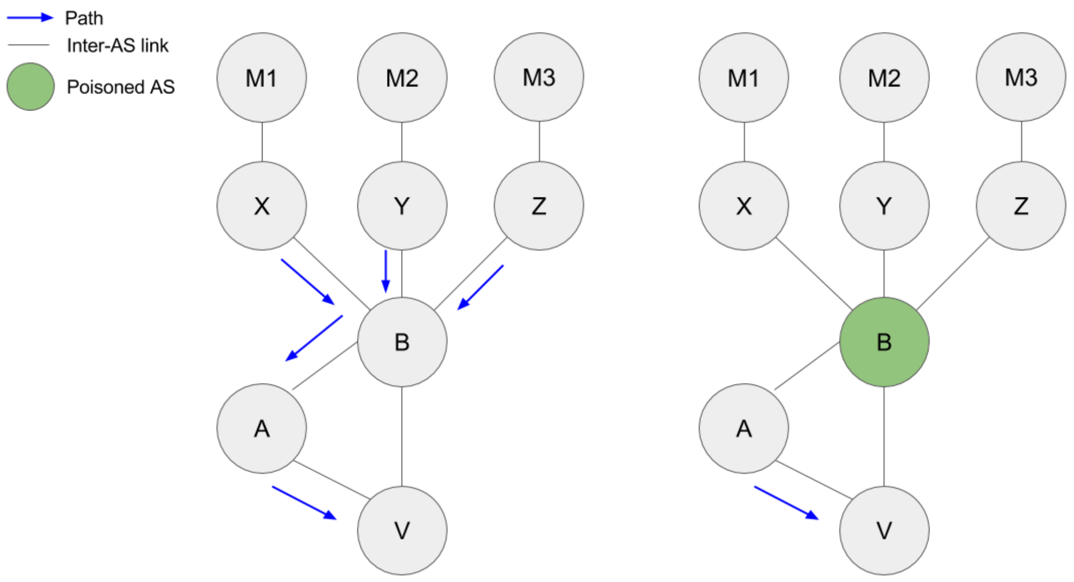


Figure 4.2: Limitation of discovering second hop neighbors using *Approach 1 - Naive*.

discover all observable second hop neighbors.

Furthermore, *Approach 1 - Naive* has deployability limitations. For most cases, this approach poisons hundreds of ASes, which is not possible due to the limitation in the length of AS path in a BGP announcement. BGP announcements with long AS path are filtered by BGP routers.

4.2.2 Depth-First Poisoning Approach (Approach 2 - Depth-First)

We devised an algorithm to overcome the discovery limitation of *Approach 1 - Naive* and maximize the number of observed second hop neighbors of an AS (Algorithm 3). The main concept of this approach is to focus on each first hop of an origin AS in turn and the second hops that are connected to that first hop. For each first hop, poison all other first hops, then poison all the observed second hop neighbors until no monitors see any route to the origin AS. This approach solves the problem illustrated in Figure 4.2: in the example, AS *B* is poisoned first and no second hops are discovered. Then, AS *A*

Algorithm 3: Discovering second hop neighbors of an AS by poisoning first hops first.

```
    Data: Victim AS $V$, Prefix $P$ (announced by $V$), a set $F$ (first of neighbors of
          $V$), a set $S_{observed}$ (observed second hop neighbors of $V$), a set of
          monitor ASes $M$
 1  for each AS X in F do
 2      Let $S'$ is a new set
 3      Let $F'$ = $F$ - AS X
 4      Announce AS path [$V$ $F'$ $V$] to poison all first hop ASes except AS X
 5      while Any M has path to P do
 6          foreach monitor in M do
 7              for AS path to P do
 8                  Add second hop neighbors of $V$ to $S'$
 9              end
10          end
11          Announce AS path [$V$ $F'$ $S'$ $V$] to poison ASes in $S'$
12      end
13      $S_{observed} = S_{observed} \cup S'$
14  end
15  return $S_{observed}$
```

is poisoned and we discover that AS X, AS Y, and AS Z are the second hop neighbors of AS V; thereby, overcoming the limitation of *Approach 1 - Naive*. We refer to this approach as *depth-first poisoning approach* or *Approach 2 - Depth-First*.

However, there are still some AS topologies where we cannot find all observable second hop neighbors using this approach. Let us consider Figure 4.3. Assume that AS S is observable. Then AS X is also observable as it exports routes to its provider AS S. However, from the monitor AS M, we do not see a path from AS S to AS V through AS X because [S N V] has a shorter path than [S X N V]. After we poison AS S according to the algorithm, we do not see any paths toward AS V (right graph in Figure 4.3) because there are no alternative routes from the monitor to AS V while avoiding the poisoned AS S. Thus, we cannot actually observe AS X, even though it is an observable second hop neighbor of AS V. Therefore, *Approach 2 - Depth-First* (Algorithm 3) allows us to discover a larger set of observed second hop neighbors of a prefix owner AS, but still

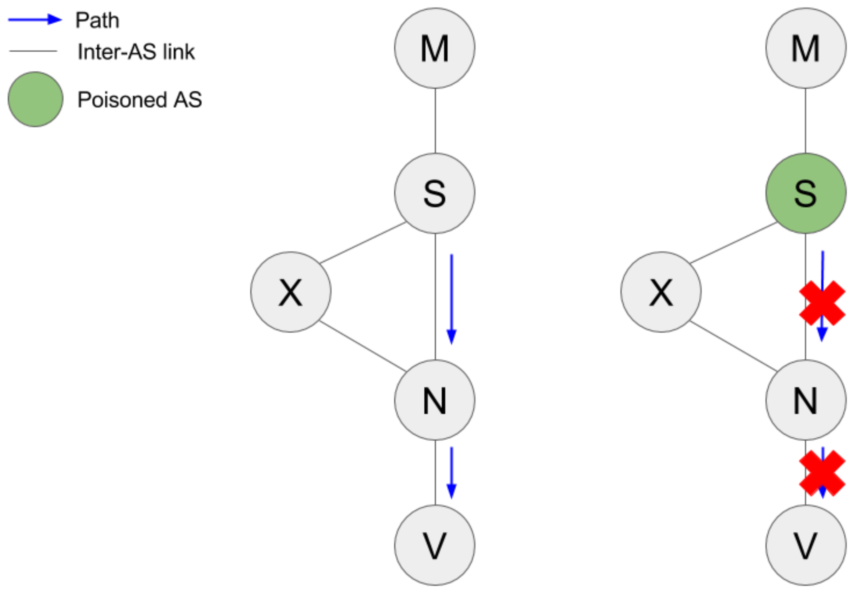


Figure 4.3: Limitation of discovering second hop neighbors using *Approach 2 - Depth-First*.

still fails to discover all observable second hop neighbors.

Furthermore, similar to *Approach 1 - Naive*, *Approach 2 - Depth-First* also has deployability limitations as this approach often poisons hundreds of ASes.

4.2.3 Poison First Hops Only Approach (Approach 3 - First Hops Only)

To overcome the practical limitations of *Approach 1 - Naive* and *Approach 2 - Depth-First*, we introduce *Approach 3 - First Hops Only* (Algorithm 4). This approach poisons all first hop neighbors of an AS excluding one, in turn, and discovers the second hop neighbors from the AS paths provided by the monitors. The number of ASes to poison reduces drastically, as this approach only poisons first hop neighbors. As a consequence, however, we cannot observe some hidden second hop neighbors. This approach is very similar to *Approach 2 - Depth-First*, but compromises discovery coverage of second hop

Algorithm 4: Discovering second hop neighbors of an AS by poisoning first hops only.

```
Data: Victim AS $V$, Prefix $P$ (announced by $V$), a set $F$ (first of neighbors of
      $V$), a set $S_{observed}$ (observed second hop neighbors of $V$), a set of
      monitor ASes $M$
1  foreach AS X in F do
2      $S'$ = new set
3      $F'$ = $F$ - AS X
4      Announce AS path [$V$ $F'$ $V$] to poison all first hop ASes except AS X
5      while Any M has path to P do
6          foreach monitor in M do
7              for AS path to P do
8                  Add second hop neighbors of $V$ to $S'$
9              end
10         end
11     end
12     $S_{observed} = S_{observed} \cup S'$
13 end
14 return $S_{observed}$
```

neighbors for deployability.

To show the deployability of this approach, we first collected 1 month period of AS path data, in April 2017, from all Route Views and RIPE RIS monitors using BGPStream. We collected a full RIB in the beginning of April 2017 and all BGP updates for the month from all monitors; therefore, this data may include transient paths. Then, we extracted the number of first hop neighbors seen by the monitors for all 56,800 origin ASes, which is one more than the number of ASes that we need to poison for each AS in order to discover second hop neighbors using *Approach 3 - First Hops Only*. In Figure 4.4, we show the CDF of the number of ASes we need to poison using this approach for all origin ASes. Assuming we set the upper bound for both the length of AS path and the size of AS set as 6 (Section 2.2), we can poison up to 24 ASes in the real world Internet. We see that we need to poison $\leq$ 24 ASes for 95% of the origin ASes. Therefore, we can use this approach for 95% of the origin ASes.

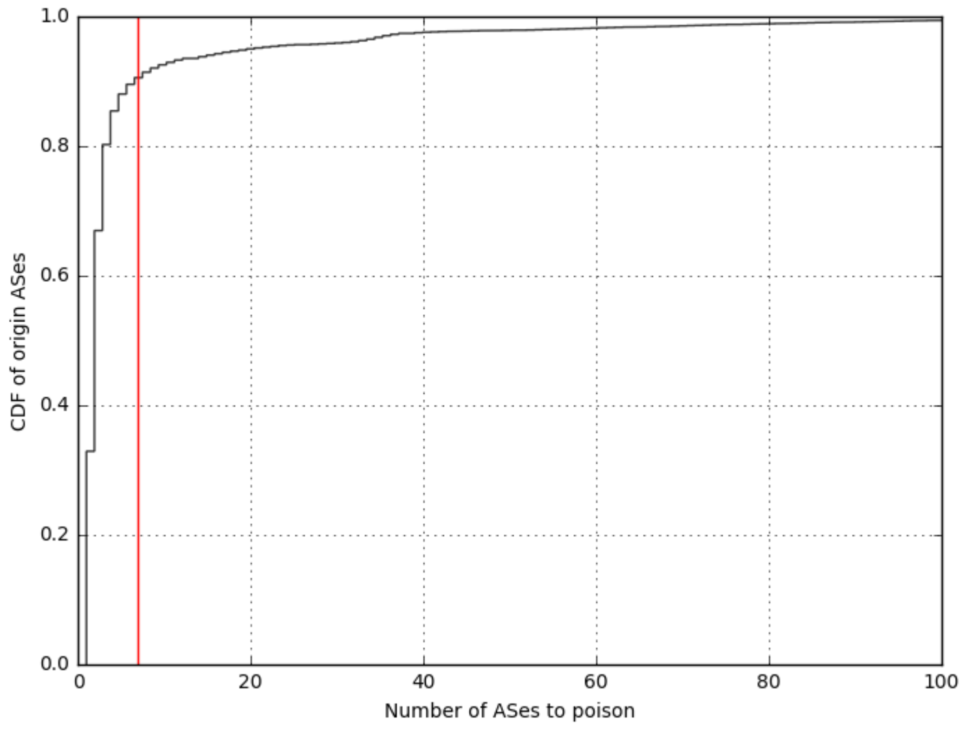


Figure 4.4: CDF of number of ASes the owner AS must poison in order to perform *Approach 3 - First Hops Only* and discover second hop neighbors for all origin ASes. Red vertical line represents the 90th percentile.

4.2.4 Poison by Divide and Conquer Approach (Approach 4 - Divide and Conquer)

Algorithm 5: Discovering second hop neighbors of an AS by poisoning by divide and conquer.

```
Data: Victim AS $V$, Prefix $P$ (announced by $V$), a set $F$ (first of neighbors of
      $V$), a set $S_{observed}$ (observed second hop neighbors of $V$), a set of
      monitor ASes $M$
foreach monitor m in M do
    for each AS X in F do
        Let $S'$ is a new set
        Let $F'$ = $F$ - AS X
        Announce AS path [$V$ $F'$ $V$] to poison all first hop ASes except AS X
        while m has path to P do
            for AS path to P from m do
                Add second hop neighbors of $V$ to $S'$
            end
            Announce AS path [$V$ $F'$ $S'$ $V$] to poison ASes in $S'$
        end
        $S_{observed} = S_{observed} \cup S'$
    end
end
return $S_{observed}$
```

We introduce *poisoning by divide and conquer approach* or *Approach 4 - Divide and Conquer*, which is deployable and has large second hop neighbor discovery coverage. In *Approach 2 - Depth-First*, we poison second hop neighbors observed by all monitors; thereby, poisoning hundreds of ASes in a BGP announcement. However, we can reduce the number of ASes to poison in an announcement by leveraging the fact that the number of second hop neighbors that each monitor discovers is small. Thus, in this approach, we focus on each monitor in turn, and perform *Approach 2 - Depth-First* using AS paths seen only by that monitor. Then, the longest announcement is calculated as follows:

$$N_{poison} = (N_{firsthop} - 1) + N_{secondhop\_max}, \tag{4.1}$$

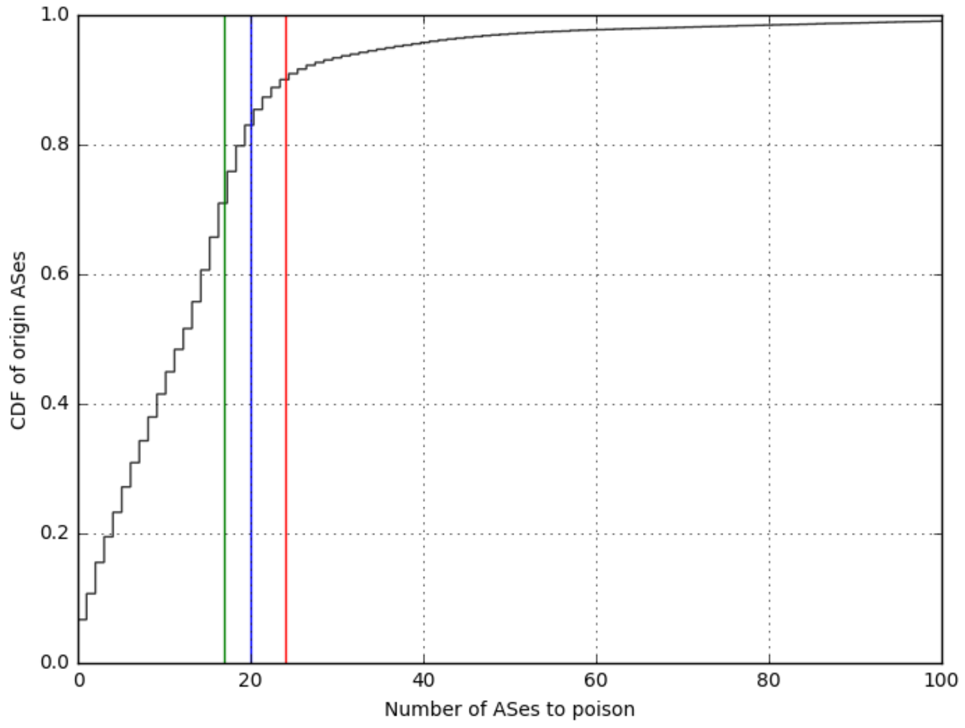


Figure 4.5: CDF of number of ASes the owner AS must poison in order to perform *Approach 4 - Divide and Conquer* and discover second hop neighbors for all origin ASes. Green, blue, and red vertical line represents the 70th, 80th, and 90th percentiles, respectively.

where N_{poison} is the number of ASes to poison using this approach, $N_{firsthop}$ is the number of first hop neighbors, and $N_{secondhop\_max}$ is the maximum number of second hop neighbors that a monitor can discover among all monitors. This dramatically reduces the number of ASes to poison, while still maintaining the large coverage in discovering second hop neighbors.

Based on the routing policy and placement of monitors, each monitor can only observe a small number of observable second hop neighbors. We show this using the real world data that we collected in Section 4.2.3. From the data, we use all AS paths seen by the monitors for all 56,800 origin ASes. We also extract the number of first hop neighbors for all origin ASes from the data. We can identify which second hop neighbors a monitor can observe using AS relationships. To infer AS relationships, we use CAIDA AS relationship data for April 2017. For a specific monitor, we check all observed AS paths. In each AS path, if the second hop neighbor is a provider of the first hop neighbor, we consider it as observable from the monitor because it will export routes to all its neighbors. If the second hop neighbor is a peer or customer of the first hop neighbor, we consider this second hop neighbor as observable if and only if the monitor is in the customer cone of the second hop neighbor. We repeat this process for all monitors and found that each monitor only observes a small number of second hop neighbors. With this result and the number of first hop neighbors, we calculate the required number of ASes to poison to perform *Approach 4 - Divide and Conquer* using Equation 4.1. Figure 4.5 shows the CDF of the required number of ASes to poison to use *Approach 4 - Divide and Conquer* for all origin ASes. We see that for 90% of the origin ASes, we need to poison 24 ASes, which is the maximum number of ASes we can poison in the Internet (Section 2.2), to discover second hop neighbors using this approach. Therefore, this approach is deployable for 90% of the ASes.

Chapter 5

Simulation Experiments to Discover Second Hop Neighbors

In this chapter, we evaluate the discovery coverage of different second hop neighbor discovery algorithms through simulation experiments: *Approach 1 - Naive*, *Approach 2 - Depth-First*, *Approach 3 - First Hops Only*, and *Approach 4 - Divide and Conquer*. We evaluate each approach by calculating the similarity of sets of observable and observed second hop neighbors for 56,800 origin ASes. To calculate the similarity of the two sets we use Jaccard's similarity measure:

$$J(S_{observable}, S_{observed}) = \frac{|S_{observable}| \cap |S_{observed}|}{|S_{observable}| \cup |S_{observed}|}, \tag{5.1}$$

where $S_{observable}$ is a set of observable second hop neighbors and $S_{observed}$ is a set of observed second hop neighbors.

For all simulation experiments, we use 2017-04-01 CAIDA AS relationship data (Section 3.1), and 258 different ASes where Route Views [oO17] and RIPE RIS [NCC17] monitors (Section 3.3) are located.

To identify observable second hop neighbors, we use the Data Concierge (Section

3.2) to find AS relationship inferences between ASes and the customer cones of ASes. For all 56,800 origin ASes, we run Algorithm 1 and identify observable second hop neighbors.

To discover observed second hop neighbors, we use the BGP simulator (Section 3.5) to simulate prefix announcements and BGP poisoning. For all 56,800 ASes, we run our second hop neighbor discovery algorithms to extract observed second hop neighbors, and calculate the similarity with observable second hop neighbors using Equation 5.1.

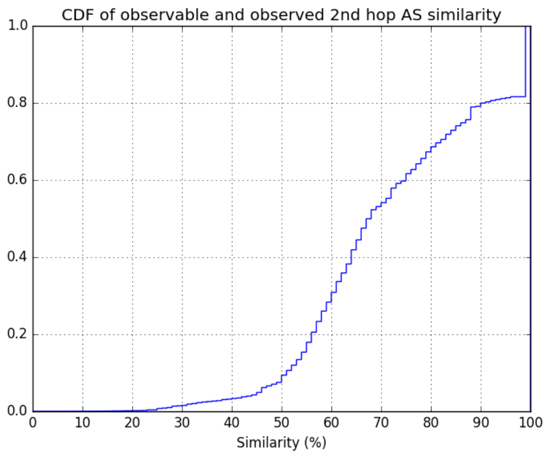
For all simulations, we found that $S_{observed} \subset S_{observable}$ for all origin ASes. Thus, the similarity between $S_{observed}$ and $S_{observable}$ can be interpreted as the percentage of $S_{observable}$ actually discovered in simulations.

5.1 Naive Poisoning Approach (Approach 1 - Naive)

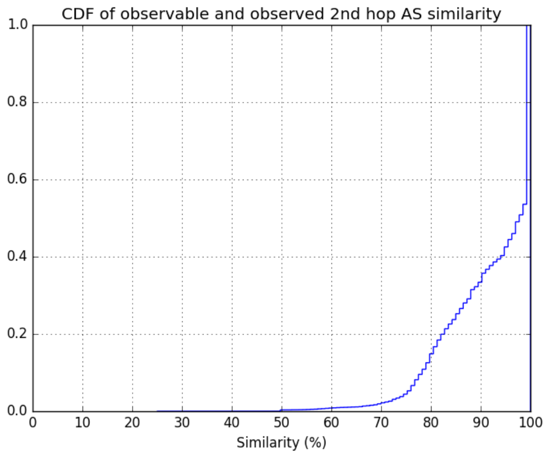
Figure 5.1(a) shows the CDF graph of similarity between observable and observed second hop neighbors for all origin ASes using *Approach 1 - Naive*. The result is disappointing. For only 37% of the origin ASes, this approach found over 80% of the observable second hop neighbors, respectively. This is largely due to the limitation of this approach mentioned in Section 4.2.1.

5.2 Depth-First Poisoning Approach (Approach 2 - Depth-First)

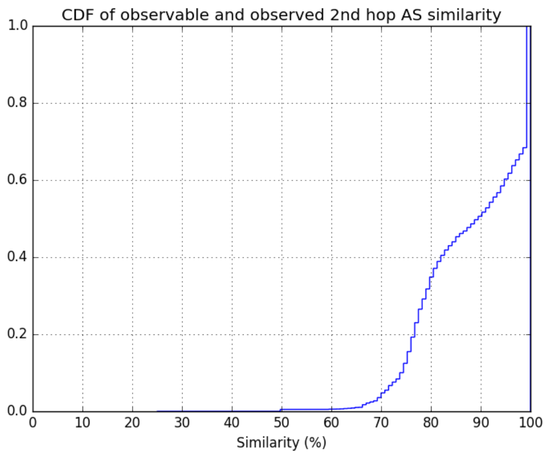
Figure 5.1(b) shows the CDF graph of similarity between observable and observed second hop neighbors for all origin ASes using *Approach 2 - Depth-First*. The result of this approach shows that in 85% of the origin ASes, it found over 80% of the observable second hop neighbors, respectively. This approach shows a great improvement from



(a)



(b)



(c)

Figure 5.1: CDF of similarity (%) of observable second hop neighbors and observed second hop neighbors discovered using (a) *Approach 1 - Naive*, (b) *Approach 2 - Depth-First*, and (c) *Approach 3 - First Hops Only*.

Approach 1 - Naive in discovering second hop neighbors. However, this approach still does not discover a complete set of observable second hop neighbors due to the limitation mentioned in Section 4.2.2.

5.3 Poison First Hops Only Approach (Approach 3 - First Hops Only)

We also simulate *Approach 3 - First Hops Only*. Since this approach compromises discovery coverage of second hop neighbors for deployability, we cannot discover as many second hop neighbors as *Approach 2 - Depth-First* using this approach. However, Figure 5.1(c) shows that this approach is still quite effective. For 65% of the origin ASes, it discovered over and 80% of observable second hop neighbors, respectively, which is a great improvement from *Approach 1 - Naive*.

5.4 Poison by Divide and Conquer Approach (Approach 4 - Divide and Conquer)

The result of the final approach, *Approach 4 - Divide and Conquer*, is identical to the result of *Approach 2 - Depth-First* (Figure 5.1(b)). Thus, this approach also has large second hop neighbor discovery coverage. However, unlike *Approach 2 - Depth-First*, this approach is also deployable because it does not require sending BGP announcements with long AS path.

Table 5.1: Summary of simulation experiments. Coverage represents percentage of observable second hop neighbors discovered. Each data cell represents that *n*% of 56,800 origin ASes discovered over 80% (Column 2) and 90% (Column 3) of its observable second hop neighbors.

	$\geq$ 80% Coverage	$\geq$ 90% Coverage
Approach 1 - Naive	37%	20%
Approach 2 - Depth-First	85%	67%
Approach 3 - First Hops Only	65%	50%
Approach 4 - Divide and Conquer	85%	67%

5.5 Summary of Simulation Experiments

Table 5.1 summarizes the evaluation of the simulation experiment. We evaluate the discovery coverage of observable second hop neighbors for 56,800 origin ASes. As shown in Table 5.1, *Approach 1 - Naive* discovers the fewest observable second hop neighbors, with only 20% and 37% of 56,800 origin ASes discovering over 90% and 80% of the observable second hop neighbors, respectively. *Approach 3 - First Hops Only* discovers more second hop neighbors than *Approach 1 - Naive*, with 65% of the origin ASes discovering over 80% of the observable second hop neighbors. Both *Approach 2 - Depth-First* and *Approach 4 - Divide and Conquer* have the largest discovery coverage, with 85% of the origin ASes showing over 80% similarity between observed and observable second hop neighbors. Moreover, *Approach 4 - Divide and Conquer* is deployable in the real world Internet (Section 4.2.4). Therefore, *Approach 4 - Divide and Conquer* is our best approach.

Chapter 6

Discussion

In this chapter, we discuss benefits and limitations of our approach for discovering second hop neighbors in BGP. Also, we discuss future work.

6.1 Benefits

We presented and evaluated various approaches to discovering second hop neighbors of an AS in BGP. We found that *Approach 4 - Divide and Conquer* discovers large sets of second hop neighbors for most origin ASes, and is also practical for use in the real world Internet. This approach offers two benefits in detecting BGP hijacking attacks: (i) the prefix owner can detect the attack by oneself, and (ii) in a timely manner.

Most impactful BGP hijacking attacks are Type-0, Type-1, and Type-2 [SKG+]. For Type-0 and Type-1 hijacking, the prefix owner can self-detect the attack. To detect Type-0 hijacking, the owner simply checks ASes that are authorized to originate its prefixes. To detect Type-1 hijacking, the owner simply checks if an AS that is one hop away from the origin AS in an AS path is one of its first hop neighbors. If not, then the owner can safely assume a Type-1 hijacking attack. However, unlike first hop neighbors, the owner lacks information about its own observable second hop neighbors. Using our approach, the owner can discover a set of its second hop neighbors, and use this set as a baseline to detect suspicious events that are potentially Type-2 hijacks. If AS *S*, that is two hops away from origin AS in an AS paths, is not in the set of second hop neighbors, then a potential Type-2 hijacking event is detected.

6.2 Limitations and Future Work

Although discovering second hop neighbors of an AS reaps great benefits, our approach has some limitations.

First, we need to send a large number of BGP announcements to discover second hop neighbors using *Approach 4 - Divide and Conquer*. The upper bound of the number of BGP announcements using this approach is:

$$N_{announcement} = N_{monitors} \times N_{poison}, \tag{6.1}$$

where $N_{announcement}$ is the number of BGP announcements, $N_{monitors}$ is the number of monitor ASes, and N_{poison} is maximum number of ASes to poison for each monitors. If $N_{monitors} = 250$ and $N_{poison} = 20$, then we need to send 5,000 announcements. Assuming that it takes 5 minutes for a BGP announcement to converge, this would take 17 days to discover second hop neighbors in the worst case.

Another limitation of our approach is that we cannot find all the observable second hop neighbors for some origin ASes. Currently, for 20% of the 56,800 origin ASes, we discover fewer than 80% of the observable second hop neighbors. Without exhaustive discovery of observable second hop neighbors for all origin ASes, we may detect many legitimate announcements as suspicious events. By improving second hop neighbor discovery coverage, we can increase the the number of correct classification of legitimate announcements, resulting in a lower false-positive rate.

We also need to standardize the upper bounds for length of an AS path and size of an AS set in a BGP announcement. Currently, network operators have different thresholds to filter out BGP messages with long AS paths or large AS sets in the path. Thus, we cannot calculate a precise number of ASes we can poison using BGP poisoning in the Internet. Standardizing the upper bounds would greatly benefit in evaluating our approaches as well as in practical use of BGP poisoning to detect BGP hijacks in the real world Internet.

Lastly, *Approach 4 - Divide and Conquer* needs rigorous evaluation in the real Internet routing before deployment. The routing policy in the Internet is much more complex than the Gao-Rexford model [GR01] that we use in the simulator. Thus, we must test and evaluate this approach rigorously in the Internet. Since there are limitations in announcing AS sets on the PEERING testbed using BIRD [BIR17] and ExaBGP [EN17], we can use a Quagga software router [Qua17] to evaluate this approach. This would allow us to perform emulation experiment of *Approach 4 - Divide and Conquer* and evaluate it in the real world Internet.

Chapter 7

Conclusion

This thesis investigates an approach for discovering second hop neighbors of an AS in BGP. To discover a complete set of observable second hop neighbors – including ASes connected through active paths, back-up paths, and alternative paths – we use a BGP poisoning technique. We devise a series of approaches to discover second hop neighbors using BGP poisoning and evaluate each approach using two criteria: (i) discovery coverage, percentage of second hop neighbors discovered, and (ii) deployment feasibility, necessary length of the AS path in a BGP announcement. We found that our *Approach 4 - Divide and Conquer* is deployable and has large second hop neighbor discovery coverage; therefore, it is the most suitable approach for discovering second hop neighbors of an AS.

We evaluated our approaches in discovering second hop neighbors by calculating the similarity of observable and observed second hop neighbors for 56,800 origin ASes. *Approach 1 - Naive* discovers the fewest of the observable second hop neighbors, with only 35% of the origin ASes discovering more than 80% of the observable second hop neighbors. *Approach 2 - First Hops First* and *Approach 4 - Divide and Conquer* have large discovery coverage, with 85% of the origin ASes discovering more than 80% of the observable second hop neighbors.

We also evaluated the deployability of each approach in terms of the necessary length of the AS path in a BGP announcement to discover second hop neighbors. We found that *Approach 3 - First Hops Only* and *Approach 4 - Divide and Conquer* need AS path length ≤ 6 for over 95% and 90% of the 56,800 origin ASes, respectively. *Approach 1 - Naive* and *Approach 2 - First Hops First* are not deployable as they need AS path length > 6 to poison hundreds of ASes for most of the origin ASes.

Finally, we built the foundations for a new approach to BGP hijacking detection. A prefix owner can easily detect BGP hijacking attacks against its own prefixes using the set of observable second hop neighbors discovered by our approach. The accuracy of hijacking detection will increase as we improve the discovery coverage of our approach.

Bibliography

[ANC$^+$15] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, Italo Cunha, Phillipa Gill, and Ethan Katz-Bassett. Investigating interdomain routing policies in the wild. *Proceedings of the 2015 Internet Measurement Conference (IMC '15)*, pages 71–77, 2015.

[BIR17] BIRD. The BIRD Internet routing daemon (BIRD). http://bird.network.cz/, 2017.

[BKN$^+$11] L. Blunk, M. Karir, Merit Network, C. Labovitz, and Deepfield Networks. Multi-threaded routing toolkit (MRT) routing information export format. Request for comments, Internet Engineering Task Force (IETF), October 2011. https://tools.ietf.org/html/rfc6396.

[BSH17] Tony Bates, Philip Smith, and Geoff Huston. CIDR report, may 2017.

[CAI17] CAIDA. BGPStream. http://bgpstream.caida.org/, 2017.

[CBP$^+$06] Lorenzo Colitti, Giuseppe Di Battista, Maurizio Patrignani, Maurizio Pizzonia, and Massimo Rimondini. Investigating prefix propagation through active BGP probing. *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC '06)*, 2006.

[CGHS16] Avichai Cohen, Yossi Gilad, Amir Herzberg, and Michael Schapira. Jump-starting BGP security with path-end validation. *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*, pages 342–355, 2016.

[Col06] Lorenzo Colitti. *Internet Topoogy Discovery Using Active Probing*. PhD thesis, University di "Roma Tre", 2006.

[EN17] Exa-Networks. ExaBGP. https://github.com/Exa-Networks/exabgp/, 2017.

[GCHS17] Yossi Gilad, Avichai Cohen, Amir Herzberg, and Michael Schapira. Are We There Yet? On RPKIs deployment and security. *The Network and Distributed System Security Symposium 2017 (NDSS '17)*, 2017.

[GR01] Lixin Gao and Jennifer Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking (TON)*, 9(6):681–692, 2001.

[HM07] Xin Hu and Z. Morley Mao. Accurate real-time identification of IP prefix hijacking. *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 3–17, 2007.

[JCC$^+$13] Umar Javed, Italo Cunha, David Choffnes, Ethan Katz-Bassett, Thomas Anderson, and Arvind Krishnamurthy. PoiRoot: Investigating the root cause of interdomain path changes. *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13)*, pages 183–194, 2013.

[JSNF$^+$16] Ed J. Scudder, Juniper Networks, R. Fernando, Cisco Systems, S. Stuart, and Google. BGP monitoring protocol (BMP). Request for comments, Internet Engineering Task Force (IETF), June 2016. https://tools.ietf.org/html/rfc7854.

[KBSC$^+$12] Ethan Katz-Bassett, Colin Scott, David R. Choffnes, Italo Cunha, Vytautas Valancius, Nick Feamster, Harsha V. Madhyastha, Thomas Anderson, and Arvind Krishnamurthy. LIFEGUARD: Practical repair of persistent route failures. *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM '12)*, pages 395–406, 2012.

[LHD$^+$13] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. AS relationships, customer cones, and validation. *Proceedings of the 2013 Internet Measurement Conference (IMC '13)*, pages 243–256, 2013.

[LMP$^+$06] Mohit Lad, Dan Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. PHAS: A prefix hijack alert system. *Proceedings of the 15th conference on USENIX Security Symposium*, 2006.

[NCC17] RIPE NCC. Routng information service (RIS). http://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/, 2017.

[OKG$^+$16] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. BGPStream: A software framework for live and historical BGP data analysis. *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*, pages 429–444, 2016.

[oO17] University of Oregon. Route views project. http://www.routeviews.org/, 2017.

[Ope17] OpenBMP. Open BGP monitoring protocol (OpenBMP) collection framework. http://openbmp.org/, 2017.

[PEE17] PEERING. PEERING: The BGP testbed. https://peering.usc.edu/, 2017.

[PK08] Alex Pilosov and Tony Kapela. Stealing the Internet: An Internet-scale man in the middle attack. *Defcon 16*, 2008.

[QGRN07] Jian Qiu, Lixin Gao, Supranamaya Ranjan, and Antonio Nucci. Detecting bogus BGP route information: Going beyond prefix hijacking. *Third International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm '07)*, 2007.

[Qua17] Quagga. Quagga routing software suite. http://nongnu.org/quagga/, 2017.

[RLH06] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). Request for comments, Network Working Group, January 2006. https://tools.ietf.org/html/rfc4271.

[SKG$^+$] Pavlos Sermpezis, Vasileios Kotronis, Petros Gigis, Xenofontas Dimitropoulos, Jae Hyun Park, Danilo Cicalese, Alistair King, and Alberto Dainotti. ARTEMIS: Neutralizing BGP hijacking within a minute. In preparation for submission to *CoNEXT '17*.

[SXW$^+$12] Xingang Shi, Yang Xiang, Zhiliang Wang, Xia Yin, and Jianping Wu. Detecting prefix hijackings in the Internet with Argus. *Proceedings of the 2012 Internet Measurement Conference (IMC '12)*, pages 15–28, 2012.

[SZC$^+$14] Brandon Schlinker, Kyriakos Zarifis, Italo Cunha, Nick Feamster, and Ethan Katz-Bassett. PEERING: An AS for us. *Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets '14)*, pages 1–7, 2014.

[ZJP$^+$07] Changxi Zheng, Lusheng Ji, Dan Pei, Jia Wang, and Paul Francis. A light-weight distributed scheme for detecting IP prefix hijacks in real-time. *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '07)*, pages 277–288, 2007.

[ZZH$^+$08] Zheng Zhang, Ying Zhang, Y. Charlie Hu, Z. Morley Mao, and Randy Bush. iSPY: Detecting IP prefix hijacking on my own. *Proceedings of the ACM SIGCOMM 2008 conference on Data communication (SIGCOMM '08)*, pages 327–338, 2008.