DISSERTATION

ROVER: A DNS-BASED METHOD TO DETECT AND PREVENT IP HIJACKS

Submitted by

Joseph E. Gersch

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2013

ABSTRACT

ROVER: A DNS-BASED METHOD TO DETECT AND PREVENT IP HIJACKS

The Border Gateway Protocol (BGP) is critical to the global internet infrastructure. Unfortunately BGP routing was designed with limited regard for security. As a result, IP route hijacking has been observed for more than 16 years. Well known incidents include a 2008 hijack of YouTube, loss of connectivity for Australia in February 2012, and an event that partially crippled Google in November 2012. Concern has been escalating as critical national infrastructure is reliant on a secure foundation for the Internet. Disruptions to military, banking, utilities, industry, and commerce can be catastrophic.

In this dissertation we propose ROVER (Route Origin VERification System), a novel and practical solution for detecting and preventing origin and sub-prefix hijacks. ROVER exploits the reverse DNS for storing route origin data and provides a fail-safe, best effort approach to authentication. This approach can be used with a variety of operational models including fully dynamic in-line BGP filtering, periodically updated authenticated route filters, and real-time notifications for network operators.

Our thesis is that ROVER systems can be deployed by a small number of institutions in an incremental fashion and still effectively thwart origin and sub-prefix IP hijacking despite non-participation by the majority of Autonomous System owners. We then present research results supporting this statement. We evaluate the effectiveness of ROVER using simulations on an Internet scale topology as well as with tests on real operational systems. Analyses include a study of IP hijack propagation patterns, effectiveness of various deployment models, critical mass requirements, and an examination of ROVER resilience and scalability.

ACKNOWLEDGEMENTS

"It's the journey, not the destination" . . . and I would never have started this journey without the encouragement of my advisor, Dan Massey and committee member Christos Papadopoulos. They started this endeavor with a friendly challenge: "You mean you're not a doctor? We thought you earned that years ago." So after a "brief" 30 year hiatus from academics, I made it a personal goal to complete this PhD before my hair went completely grey.

First, I have to thank my advisor and committee. Dan was the focal point for the discussions, designs and the many ideas we kicked around together and with others. Dan and Christos were content experts: invaluable for their support, guidance, close involvement with the technological issues, and in promoting ROVER as an important concept at various forums. Michelle Strout and Stephen Hayne provided many valuable insights from a computer science and business school perspective.

I also appreciate the support from my colleagues at Secure64 Software Corporation. Denny Georg, Steve Goodbarn and Bill Worley are the executives who supported and encouraged my ideas. All of the engineers were good sounding boards for ideas. A special thanks to David Roth for his reviews and critiques of papers and this dissertation.

Finally, I must thank the most important person of all, my wife Karen. She put up with the many long nights and weekends while I wrote programs, tested new concepts, and wrote papers and presentations. It consumed a lot of time in which we could have been doing other things together, but she encouraged me to continue because she saw my passion for the subject and had the patience to see me through to the end.

It really does take a village to raise a child, and it probably takes even more to raise a PhD. To all of those who helped but haven't been listed by name, I say "Thank you".

DEDICATION

To my wife, Karen, with love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ARIN	American Registry for Internet Numbers
AS	Autonomous System
BGP	Border Gateway Protocol
CIDR	Classless Inter-Domain Routing
DoS	Denial of Service
DDoS	Distributed Denial of Service
DNS	Domain Name System
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
IETF	Internet Engineering Task Force
IETF	Internet Engineering Task Force
IRR	Internet Routing Registry
ISP	Internet Service Provider
PKI	Public Key Infrastructure
RIPE	Résaux IP Européens Network Coordination Centre
ROA	Route Origin Authorizations
RIR	Regional Internet Registry
RPKI	Resource Public Key Infrastructure
QPS	Queries Per Second
RADb	Routing Assets Database
RIB	Routing Information Base
RLOCK	Route Lock

SBGP	Secure Border Gateway Protocol
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SRO	Secure Route Origin
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol over Internet Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

In April of 1997 a small ISP in Florida crippled the Internet for two hours due to a router misconfiguration [5]. Their unintentional error caused a router to send incorrect BGP announcements to neighboring routers which naively accepted them and continued to propagate them onwards. The erroneous announcements spread widely resulting in a flood of traffic to the ISP that overwhelmed their capacity and triggered the outage. Even though it was an accident, this was the first documented case of IP hijacking.

In 2008, a Pakistan ISP performed an intentional hijack of a YouTube address block, attempting to censor content in Pakistan only [4]. Unfortunately the ISP's route filters incorrectly leaked the route to neighboring routers which propagated the announcement across the world. This effectively took YouTube offline for 45 minutes.

As recently as February 2012 millions of people across the entire continent of Australia were taken offline for 45 minutes due to a BGP filter error that created a routing valley [45]. Tier-1 traffic from Telstra and Optus were mistakenly routed through a much smaller regional ISP, overwhelming their capacity and causing the outage. And in November 2012, Google was partially crippled when a small Indonesian ISP announced the wrong address space.

These examples, spanning sixteen years of internet history, demonstrate the severity of IP hijacking. Attacks both large and small continue to this day. Some have consequences large enough to be publicized in the popular press.

1.1. *Problem Statement*

IP Hijacks can occur by accidental misconfiguration or by malicious intent. When malicious, a hijack can be used to blackhole address blocks, effectively removing organizations from the Internet. They can also be used to intercept traffic or impersonate a victim's website. Organizations are mostly defenseless. Firewalls and Intrusion Protection Systems can do nothing to prevent the attack; the vulnerability is at a much lower level, within the fundamental routing mechanism of the Internet itself.

Recognized as a problem for many years, IP hijacking is becoming even more of a hot topic as internet organizations and governments voice their concerns. The U.S. Department of Homeland Security views IP hijacking as a fundamental cyber-security topic. FCC Chairman Julius Genachowski recently called IP hijacking one of the top three cyber-security threats, the others being bot-nets and DNS hacking [29].

The root cause of this problem is that, unfortunately, BGP routing was designed with only limited regard for security. To address this shortcoming, many academic papers have been published over the past years that propose a variety of solutions for detecting and preventing IP hijacks. IETF working groups have also developed proposals and testbeds for solutions such as S*BGP (*Secure BGP* and its variants). Their current work focuses on RPKI (*Resource Public Key Infrastructure*), a method to publish authorized route origins in a cryptographically signed certificate hierarchy. These approaches will be summarized in Chapter 2.

Despite the potentially serious consequences of IP hijacks, none of these solutions have been widely deployed. Reasons usually cited are cost, complexity and lack of consensus. Many ISPs do not view their risk as large enough to warrant investment. Nevertheless, a number of organizations including critical national infrastructure have a very high risk and require a working, deployed solution. This creates a dilemma in that critical sites need to be protected despite lack of action from the majority.

1.2. *Thesis Statement and Contributions*

Given the context and historical background, this dissertation focuses on the problem of origin and sub-prefix hijacks. Our thesis statement is *"A system based on the existing DNS infrastructure can be deployed by a small number of institutions in an incremental fashion and still effectively thwart origin and sub-prefix IP hijacking despite non-participation by the majority of Autonomous System owners."*

To prove this statement a number of novel concepts and tools had to be developed. These are listed in section 8.1 and include the ROVER domain naming convention and validation algorithm. A live internet testbed was developed and a global internet simulation tool was written to perform various research experiments. Graphical visualization programs were also written to help build insights regarding IP attacks and defenses. New internet topology metrics, *depth* and *reach* are introduced. Finally, the research results and measurements were performed in support of the thesis statement.

1.3. *Research Objectives*

This dissertation proposes a novel invention called "ROVER", an acronym for *Route Origin VERification*. ROVER is small set of mechanisms that enable practical solutions for the detection and prevention of origin and sub-prefix hijacks. An conceptual overview of ROVER is depicted in Figure 1.1.

ROVER exploits the existing reverse DNS infrastructure as a means for storing and retrieving owner-authorized route origin data. These DNS data are cryptographically protected from fraud

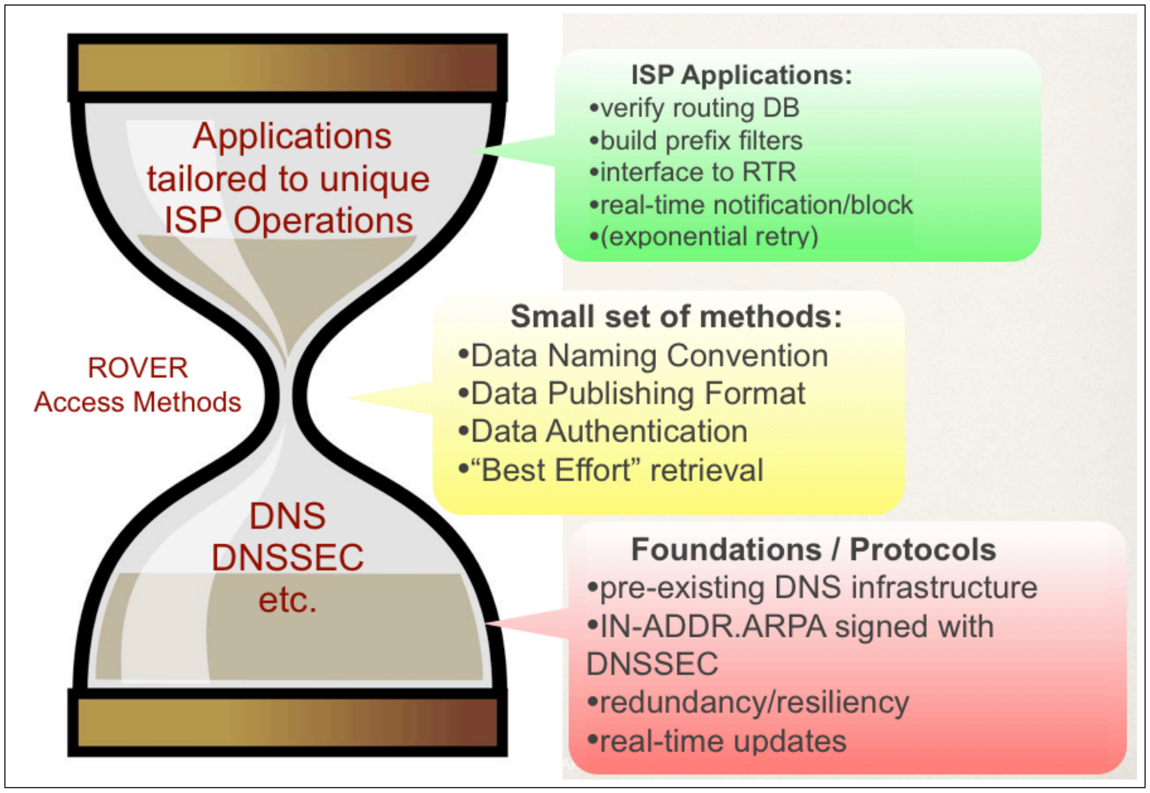


Figure 1.1. *ROVER Design Model*

by DNSSEC signing and validation mechanisms. The ROVER algorithm provides a fail-safe, best effort approach to route origin authentication. ROVER mechanisms can be incorporated into a variety of operational models including fully dynamic in-line BGP filtering, periodically updated authenticated route filters, and a hijack detection system that issues real-time notifications for network operators.

A feasibility study of ROVER was performed by developing a live Internet testbed employing a web-based user interface. Participants in the testbed could publish route origins in the reverse-DNS using a built-in provisioning system. As more and more data was published, the testbed was used for measuring query performance, assessing architectural tradeoffs, finding and fixing design flaws, and other useful experiments. The testbed also implemented an IP hijack detector which compared a real-time stream of BGP announcements to data retrieved from the DNS.

The next step was to perform analytic research. Several questions were posed to build the case for ROVER. The most obvious high-level questions are “does it really work?”, “how well does it work?” and “what are its issues and limits?” Other questions include:

- What system measurements are relevant?
- Is there a critical mass threshold required before ROVER can be considered globally useful? How many detectors are necessary? Where should they be placed?
- What is the impact to the DNS system in terms of load, fanout, robustness, failures, real-time capable, etc. Does DNS cause a circular dependency with BGP?
- What are the threat models to ROVER. Can the vulnerabilities be prevented or mitigated?
- Can a reliable system be built when ROVER’s core dependency is based on a potentially unreliable DNS?

These questions were investigated with a variety of tools including simulation, the construction of a real-world ROVER testbed, DNS measurements, etc. To provide structure to the research, a set of four research goals were defined and pursued for this dissertation.

(1) Characterize IP Hijack propagation.

(2) Measure the effectiveness of the proposed solution.

(3) Characterize the limits and resiliency of the proposed solution.

(4) Measure the scalability and impact of the proposed solution on the Global DNS.

This research created some interesting and occasionally surprising results. We were able to formulate several new metrics that correlate attack vulnerability with a router’s position in the Internet topology. We determined that a deployment strategy for protecting Internet routing that only places filters at the Tier-1 ISPs does not work. We found techniques to mitigate DDoS and other attack vectors that attempt to thwart the ROVER operations. We also determined that the global DNS infrastructure has the capacity and scalability to support the load posed by ROVER.

The remainder of this dissertation is organized as follows. Chapter 2 gives a brief overview of BGP essentials and a survey of prior work related to IP hijacks. Chapter 3 describes the functional and operational design of ROVER. Chapters 4, 5, 6, and 7 discuss the research conducted and results obtained. Finally we summarize conclusions and contributions in Chapter 8 and list publications and presentations in Chapter 9.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter surveys current knowledge of IP hijacking: how it occurs, its effects, and what previous solutions have been proposed.

These topics are organized as follows. Section 2.1 reviews the basics of inter-domain routing using the Border Gateway Protocol. Section 2.2 surveys vulnerabilities in routers and in the BGP protocol itself, then explores a range of IP hijacking methods, limitations, and evasion techniques. Section 2.3 examines a taxonomy of detection and mitigation techniques utilizing information from the router control and data planes. We conclude in Section 2.4 with an examination of hijacking importance in terms of magnitude, effects, and whether any of the proposed detection or mitigation techniques is likely to be widely deployed relative to cost and benefits.

2.1. *Border Gateway Protocol*

2.1.1. *BGP Fundamentals.* This section describes only the aspects of BGP and related terminology [42] that are relevant to IP hijacking and its detection. Comprehensive information on BGP is otherwise widely available.

The Internet is comprised of a set of independent sub-networks called *Autonomous Systems (AS).* There are currently over 60,000 ASes distributed around the world. Each AS may be assigned or allocated one or more blocks of IP addresses designated by a CIDR prefix (e.g. 129.82/16). The AS owning the prefix is referred to as the “origin AS”.

Routing tables are constructed as BGP UPDATE and WITHDRAW messages are received by a router from its adjacent neighbors. An UPDATE message specifies a prefix and a path vector of sequential AS numbers specifying the route back to the origin AS. BGP avoids loops by having each router examine the path and ignoring an announcement if it see that its own AS number is present. Router data structures include:

- RIB: Routing Information Base
  - Adj-RIBs-In: raw routing data sent by neighbors.
  - Loc-RIB: actual route chosen from Adj-RIBs-In.
  - Adj-RIBs-Out: routing data to be sent to neighbors.
- FIB: Forwarding Information Base – used to forward a packet to its next hop.

In a nutshell, BGP route announcements are propagated as follows: the origin AS announces a prefix and its AS number to its neighboring ASes. If the new route is "better" than those in a router's existing tables, the router will prepend its own AS number to the path and propagate the new route to its set of neighbors. This continues until the route is fully propagated and globally stable. An example route propagation is illustrated in Figure 2.1.

High-speed packet forwarding is controlled by routing entries in the FIB. Note, however, that if a packet's IP address matches multiple prefixes stored in the FIB, the *most specific* prefix always wins. As an example, consider the case where a FIB contains routes for both 129.82/16 as well as 129.82.0/18. A packet destined for 129.82.0.10 matches both, but the most specific prefix, 129.82.0/18 determines the chosen route.

Choosing the "best route" for insertion into the FIB is a key factor in BGP routing (and in BGP attacks). The decision algorithm checks a sequence of attributes in the following strict order when comparing paths:

(1) WEIGHT

(2) LOCAL_PREF

(3) NETWORK or AGGREGATE

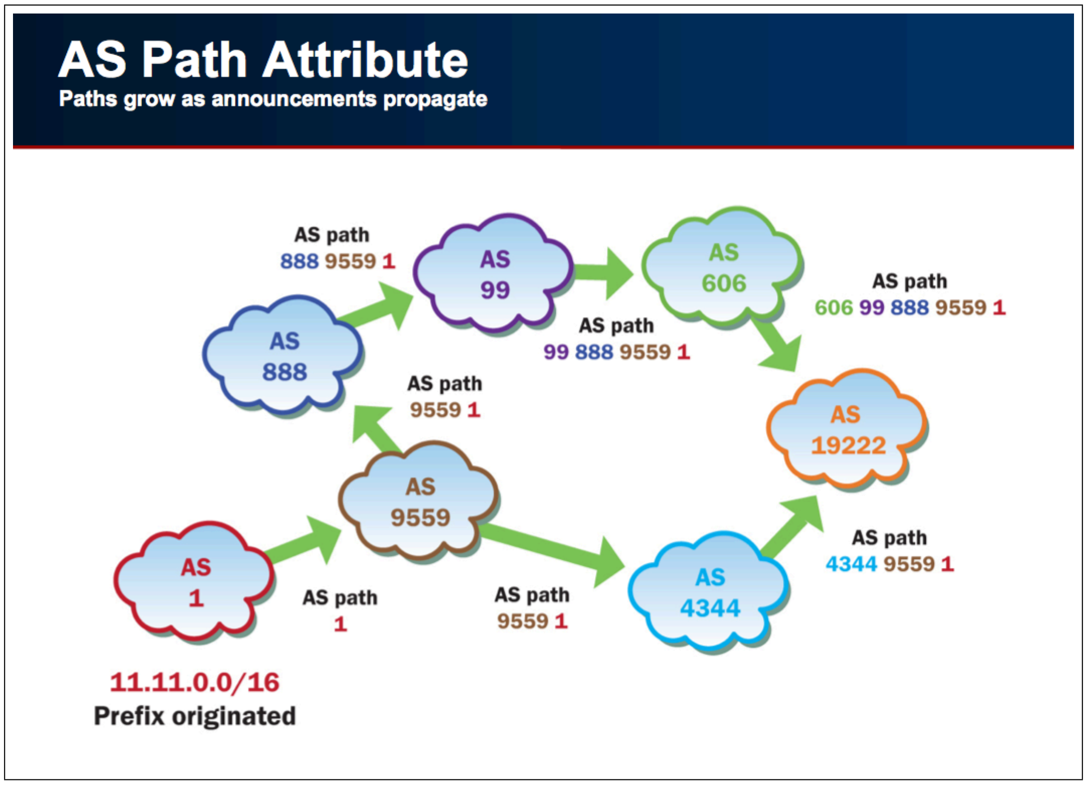


Figure 2.1. *AS Path Propagation (from [19])*

(4) Shortest AS_PATH

(5) followed by 9 other attributes

The two most important of these attributes with respect to IP hijacking are LOCAL_PREF and SHORTEST AS_PATH.

2.1.2. *AS relationships.* The communication between one AS and another is often classified according the their business relationship:

- *Transit Provider*: is paid by a customer to provide connectivity to the internet
- *Customer*: connects to a provider and pays for data transfer
- *Peer*: neighbors that agree to interconnect and exchange routing information; typically no money changes hands. Note that ISP business relationships are typically confidential. A peer relationship may never be disclosed to the general public.

- *Sibling*: Peer ASes typically owned or managed by the same organization, perhaps because of a business acquisition.

The economics of an AS relationship imply that an ISP would prefer to route traffic through a peer if possible, rather than pay a provider for transit. Also, routes received from a peer should not be advertised by an AS because this would inadvertently cause the peer to transit traffic it shouldn't be responsible for. Routing valleys should be avoided (provider to customer to provider) because this can cause horrible traffic problems due to overload. These behaviors are managed by BGP routing policies.

Another AS classification used by many of the research papers describe an ISP as being a tier-1, 2 or 3 provider. This is illustrated in Figure 2.2 and defined as follows:.

- Global Tier-1: transit-free; connect to entire net without purchasing transit from anyone; they only peer to other tier-1s; approximately 15-20 ISPs
- Regional Tier-1: may purchase transit, but cover a whole country or region without peering; approximately 200 ISPs
- Tier-2: purchases some transit and does peering with other networks/exchanges; there are approximately 3500 ISPs in the US, 100,000 ISPs world-wide, plus the large enterprises who run their own tier-2 networks (perhaps 5000)
- Tier-3: A network that solely purchases transit from other networks to reach the Internet

2.1.3. *Routing Policy.* BGP by itself has no knowledge of a customer, provider or peer relationship. These relationships are managed by defining a router policy that controls the behavior of the autonomous system. Policies can be applied individually to internal and external routes and can control the propagation of routes.

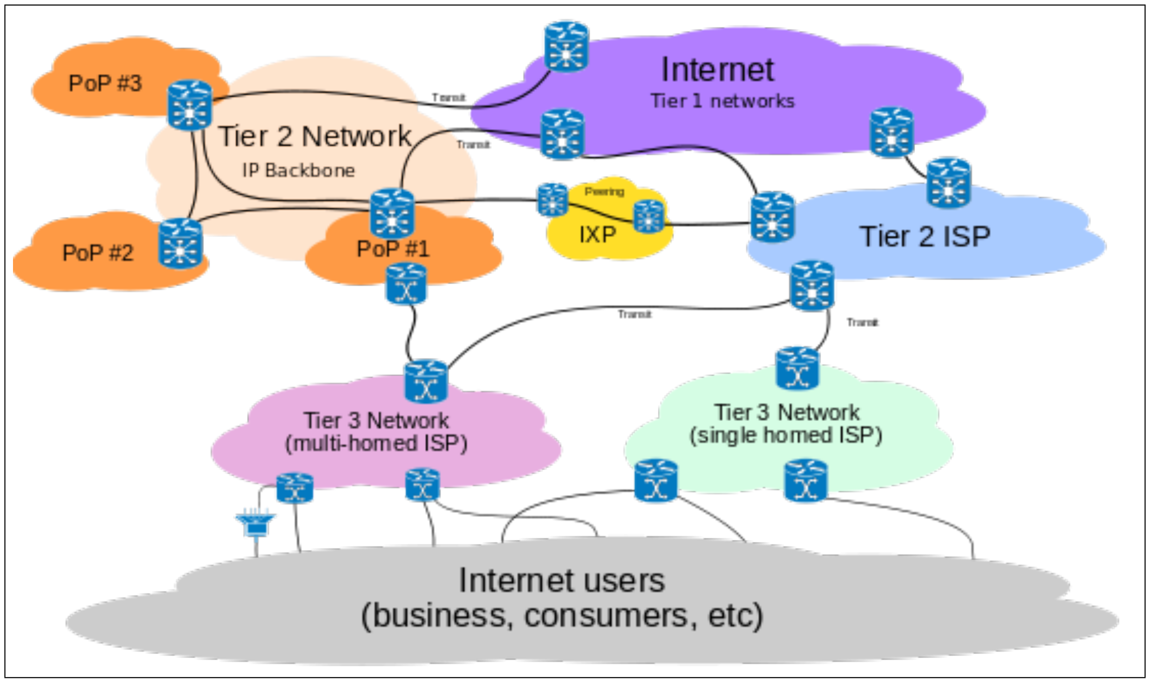


Figure 2.2. *Internet Tiers (from wikipedia)*

A Routing policy is implemented by applying route filters and route maps. Route filters control which routes are to be advertised (egress filter) and which can be received (ingress filter). Route maps are a set of sequential rules that control traffic flow and routing behavior. A route map verifies whether a route matches the rule, permit or deny the route, and execute commands that adjust attributes of a route. Attributes include variables such as LOCAL_PREF, access-lists, AS-path prepending, multi-exit discriminator, and community strings to define further behavior and filtering by other ISPs.

The AS relationships described earlier are managed by setting appropriate router policies. Traffic flow is handled by setting LOCAL_PREF. Most ISPs assign higher LOCAL_PREF values to customer-routes than to peer-routes than to provider-routes. Route propagation between ASes is controlled by egress filters:

- *customer to provider*: export all routes except those of its providers and peers
- *provider to customer*: export all routes

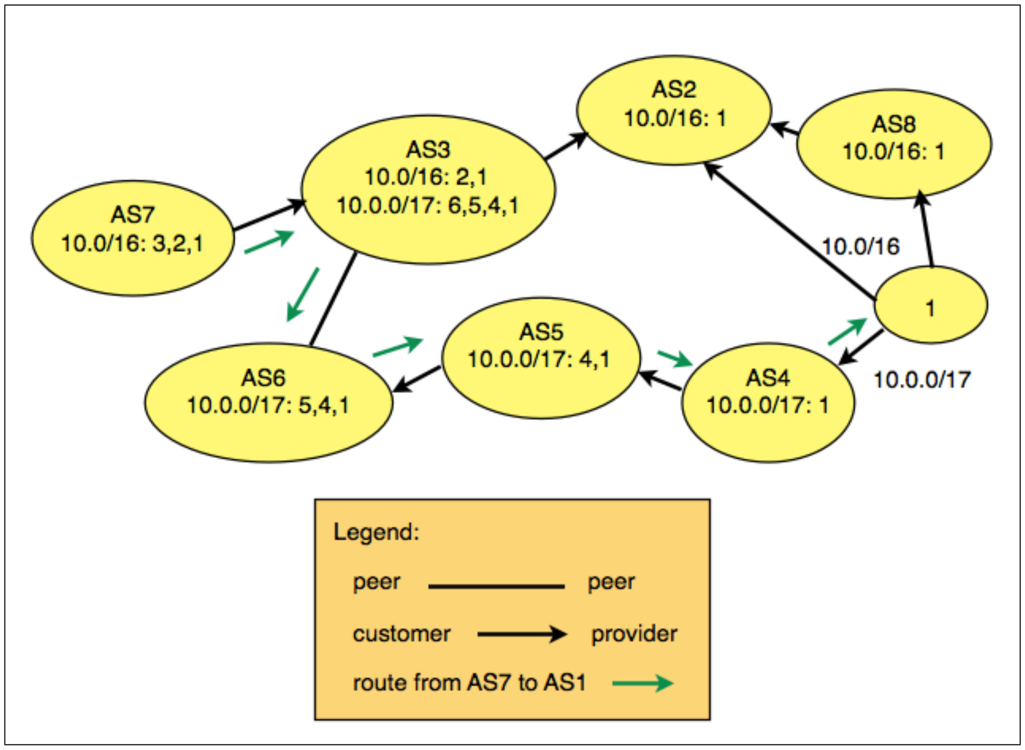


Figure 2.3. *Example: AS Path Selection*

- *peer to peer*: export customer and sibling routes only
- *siblings to sibling* export all routes, behave as a single consolidated AS

Filtering is an extremely important part of routing policy. Unfortunately, not everyone sets filters, or sets them correctly. Errors can cause IP hijacks to propagate widely, or for misconfigurations to propagate via a “route leak”.

To show the interplay between path selection, propagation, and the rule of most-specific-prefix, consider the example illustrated in Figure 2.3.

AS1 originates a prefix, but for traffic control reasons, sends a /16 to both AS2 and AS8 and sends a more specific /17 to AS4. Egress filters control the route propagations. The first interesting situation occurs at AS2. It receives routes from AS1 and a competing route from AS8. The shortest path rule wins. The next interesting situation is at AS3. It receives a /16 from AS2 and a /17 from

AS6, but it is not allowed to propagate the /17 to AS7 because it got it from a peer. AS7 only receives the /16 route.

Given the topology, one could expect that a data packet sent from AS7 to AS1 would follow the path from 7 through 3 to 2 to 1. What actually happens is different because all routing decisions are made locally. The packet arrives at AS3 which has a more specific prefix in its FIB, so it immediately forwards it to AS6. The actual data path is AS7, AS3, AS6, AS5, AS4, AS1 – a path that is suboptimal in length, but logical in terms of routing policy. A similar example could be constructed using LOCAL_PREF.

2.2. *BGP Attack Techniques*

BGP was designed at at time when there was implied trust between the various actors. That world is long gone and hostile actions are now a fact of life. With very few exceptions, IP hijacks are serious attacks generally used to

- Blackhole – routing traffic away from a victim and dropping the packets, effectively making the victim inaccessible.
- Impersonate – instead of dropping the traffic, imitate their websites or other servers. Impersonating a major DNS server (w/o DNSSEC) would extend the attack to a much wider area of the net.
- Eavesdrop – listen in on all traffic, perhaps as a man-in-the-middle.
- Spam – send spam email while temporarily hiding in someone else's address space to avoid detection.

However, there is at least one benevolent IP poisoning attack described in the paper by Katz-Bassett *et al* [26] . This will be described in Section 2.2.5 regarding Machiavellian Routing. Their paper also cites references that use benevolent poisoning to uncover hidden network topology and assess the prevalence of default routes.

This section summarizes the BGP threat model including router platform vulnerabilities, IP hijacking and evasion techniques, factors that limit the successful launch of a hijack, and the frequency and duration of observed attacks. Many of these issues are described in the 2010 paper by Butler *et al* [5] and and the 2011 paper by Huston *et a*l [24]. Both of these papers are comprehensive survey articles, similar in scope, and include very extensive bibliographies. Other papers are cited as necessary.

2.2.1. *Router Platform Vulnerabilities.* Launching an IP hijack requires an attacker to directly control a router or to compromise someone else's router located at a point that will impact the intended victim. Routers are vulnerable. Many administrators have never changed their router's default password so taking control is as simple as logging on. Otherwise Standard interfaces such as SSH and Telnet have vulnerabilities that can be exploited.

The next concern is that BGP routers use TCP sessions to communicate with each other. This allows for another set of vulnerabilities.

(1) Attackers can eavesdrop, alter or inject TCP messages. In fact, an entire TCP session can be hijacked by forcing router resets and guessing sequence numbers.
(2) Denial of Service attacks (DOS) can be launched in many ways including SYN floods, data floods, and TCP reset attacks. A MITM attacker can withhold traffic until the BGP KEEP-ALIVE fails causing complete router table retransmission, effecting a DOS attack.
(3) A TCP replay attack can be used to suppress a route across the internet (blackholing the victim) by simply repeating a WITHDRAW / UPDATE sequence often enough to trigger the BGP route flap damping mechanism.

(4) A man-in-the-middle attack can delay packets so that a router falls behind. THe MITM can listen in and forge messages, guess sequence numbers and inject harmful messages.

IPSEC is one method to protect against TCP attacks; however few people implement IPSEC. MD5 with a shared secret key is more generally used and can verify the integrity and source of the BGP message. Digital signatures verify the payload hasn't been altered (it can be wrong, but it hasn't been altered!) However MD5 has a heavier computation load which creates a larger opportunity for DOS attacks. Key rollover and scaling is also an issue. But MD5 is better than nothing.

Another TCP defense is to use the Generalized TTL Security Mechanism (GTSM). In GTSM, a BGP receiver will discard all packets whose TCP TTL is less than a certain threshold, e.g. 255. This ensures that a sending router must be within a certain radius of the receiver. More distant Attack Routers will have a TTL less than 254 and thereby be ignored.

2.2.2. *BGP Protocol Vulnerabilities.* Huston [24] is quite specific in calling out the threats to the BPG protocol:

- no mechanism to verify integrity, source and freshness of BGP messages,
- no mechanism to verify prefix/origin,
- no mechanism to verify attributes in UPDATE,
- no mechanism to verify local cache RIB is consistent with current state of forwarding table.

Given this lack of verification, an attacker is able to craft a variety of invalid BGP messages resulting in various types of IP hijacks. These are categorized as follows (list compiled from multiple sources [5, 22], etc.):

(1) *Origin Hijack*: also known as a *prefix hijack* occurs when an AS announces itself as the origin of someone else's prefix. The bogus path will propagate depending on policy and shortest-path rules. Affected ASes are now routed to the the hijacker, making the legitimate owner unreachable. An origin hijack results in Multiple Origin AS (MOAS) conflicts [53] unless the prefix being stolen is unused space. A sample origin hijack is illustrated in Figure 2.4.

(2) *Sub-prefix Hijack*: is similar to an origin hijack except that the prefix announced is a smaller subnet of an existing prefix. This hijack will propagate widely because more specific routes are chosen by traffic forwarding. A sub-prefix hijack does not create a MOAS conflict.

(3) *Next-hop Attack*: rather than steal the origin, the attacker places itself as the next-to-last hop in the path, thereby eliminating a MOAS conflict and making detection harder. This can be done for both prefix and sub-prefix hijacks.

(4) *Path spoofing*: In this case the hijacking AS could be placed later in the path, making it harder to detect. Most paths have an average length less than 4, however, making this more difficult to launch; indeed, this is the least observed attack method. Spoofing attacks can also inflate path lengths by repeating an AS number to make a path less desirable.

(5) *Stealthy Attack*: is carefully crafted path spoofing where the attacking AS is moved up or down in the path. McArthur and Guirguis [31] analyzed how an attacker could avoid detection by deflecting only a small portion of the traffic to an AS so that the bulk of the traffic still arrives. They also demonstrated how *trace route* could be tricked into returning spoofed TTL's, making stealthy attacks even harder to detect.

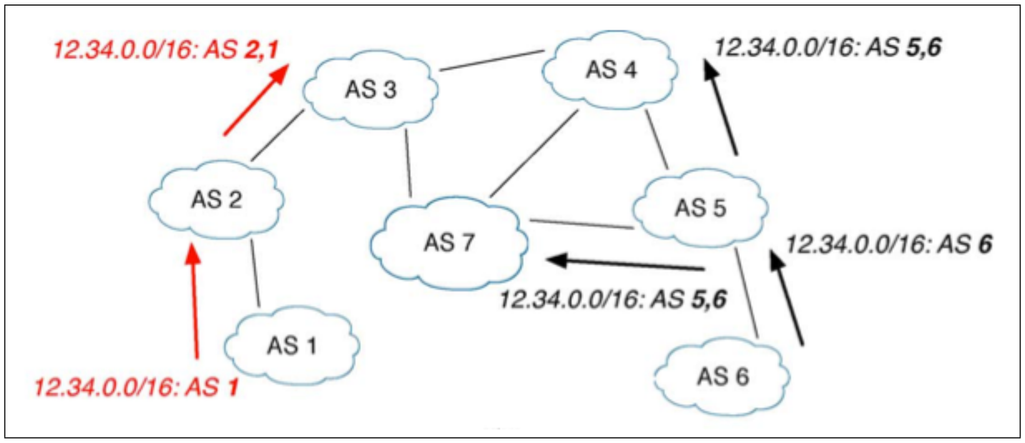


Figure 2.4. *Origin Hijack (from [5])*

(6) *Man-in-the-Middle*: This attack was made famous at the 2008 DEFCON hacker's conference in a presentation called "stealing the internet" by Pilosov and Kapela where the authors showed how it was possible to alter or simply eavesdrop on intercept traffic without being detected. The attack was later written up by Renesys [19].

The MITM attack occurs in steps in which *trace route* was used to observe paths as a method to carefully choose a path back to the AS which would remain unaltered. Attacks were made from a multi-homed AS to other routing paths, and normal BGP loop prevention prevented the propagation of these bad path from the "safe reply path". The attackers also silently increased the TTLs of packets in transit to hide the presence of the attacking AS.

An example of the victim AS with its unaltered reply path, the intercepting AS, and hijacked paths are illustrated in Figure 2.5.

(7) *Data Plane Hijack*: In this case the attacker's router is already in the routing path. No alterations are made in the control plane. Instead the attacker simply drops, listens in, or alters the packets in transit.

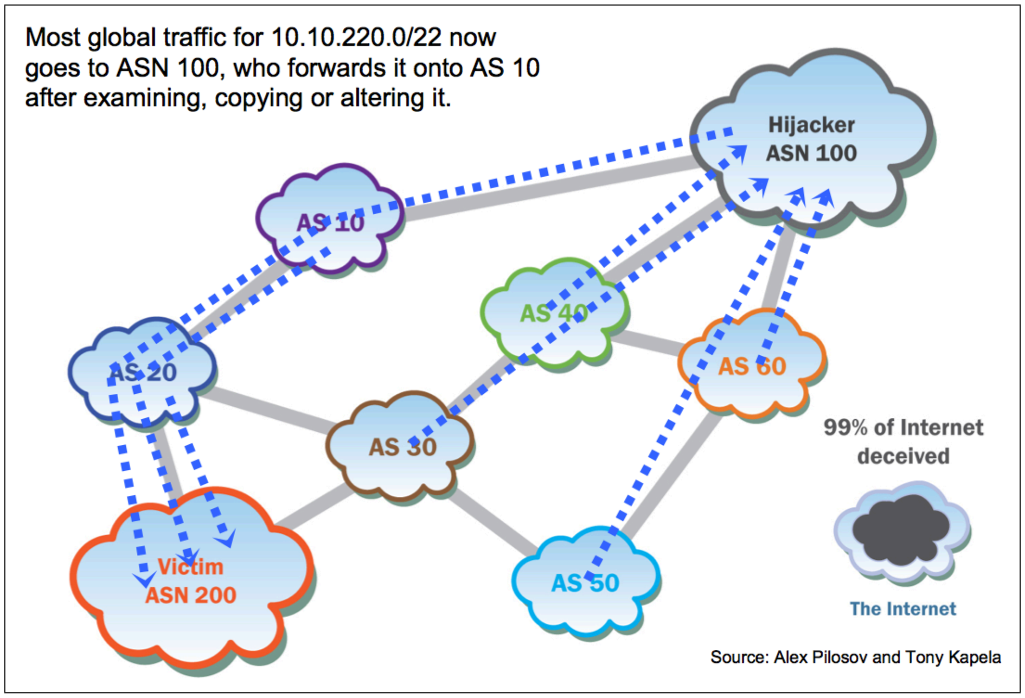
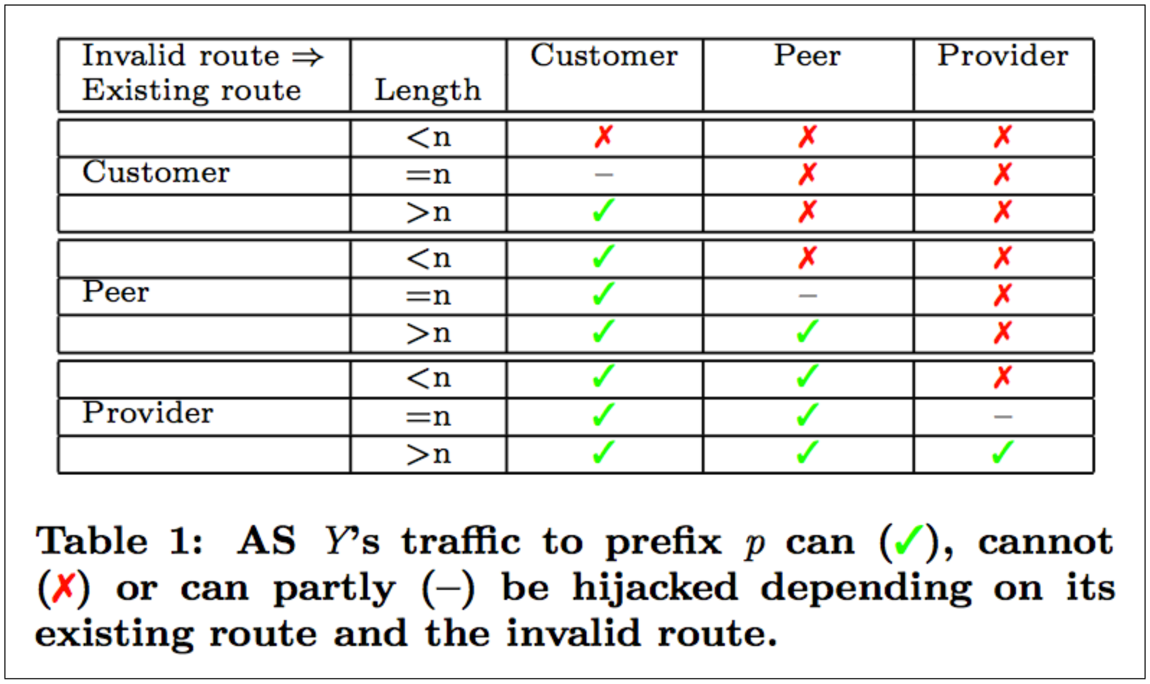

Most global traffic for 10.10.220.0/22 now goes to ASN 100, who forwards it onto AS 10 after examining, copying or altering it.
Hijacker ASN 100
AS 10
AS 40
AS 20
AS 60
AS 30
99% of Internet deceived
Victim ASN 200
AS 50
The Internet
Source: Alex Pilosov and Tony Kapela

Figure 2.5. *Man-in-the-Middle Attack (from [19])*

(8) *Route Valley*: Strictly speaking this is more of a misconfiguration than an attack, but the end result could be a drastic overload at an AS. A valley is typically seen as Provider-Customer-Provider path. More specifically Gao defines it as *after a provider-to-customer edge or a peer-to-peer edge, the AS path can not traverse customer-to-provider edges or another peer-to-peer edge* [8].

2.2.3. *Probability of Launching a Successful Attack.* Launching an IP hijack doesn't mean it will necessarily succeed. Neighboring routers might have policies that filter out the attack or choose an alternative path with other preferential attributes.

The research by Ballani *et al* [1] examined a number of factors that determine the fate of an attack. Figure 2.6 illustrates 9 cases where a hijack attempt will or will not be accepted by a neighbor based on peer-customer-provider relationships.



Invalid route ⇒ Existing route	Length	Customer	Peer	Provider
Customer	<n	✗	✗	✗
	=n	–	✗	✗
	>n	✓	✗	✗
Peer	<n	✓	✗	✗
	=n	✓	–	✗
	>n	✓	✓	✗
Provider	<n	✓	✓	✗
	=n	✓	✓	–
	>n	✓	✓	✓

Table 1: AS *Y*'s traffic to prefix *p* can (✓), cannot (✗) or can partly (–) be hijacked depending on its existing route and the invalid route.

Figure 2.6. *Factors affecting bogus route propagation (from [1])*

The authors' analysis showed that a tier-1 AS can, on average, hijack traffic for a prefix from another tier-1 AS with 70-75% probability. Hijacking from any AS had a success probability ranging from 38-63%. They also examined interception (eavesdropping) attacks which require maintaining a safe route back to the victim. In these cases the probability for any AS to intercept traffic ranged from 29-48% for any AS, and 52-79% for tier-1 Ases. They verified their projections against known events and although some cases gave over- or under-estimates, most of the estimates were accurate. Finally, the authors performed a study to detect intercepting route attacks, but they had a null result; they could neither detect nor definitively prove that there were no ongoing interception attacks.

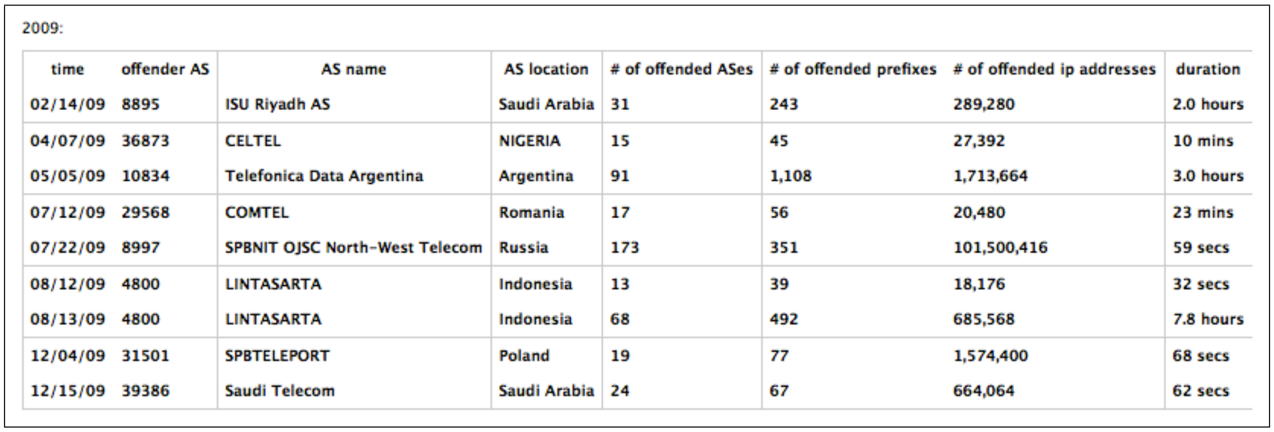
2.2.4. *Attack Frequency, Duration, and Extent.* Only a few of the papers on IP hijacking quantify the magnitude of the problem. Most of the papers simply cite an example high profile attack but give no further data (e.g. the Pakistan/YouTube hijack). The papers that did provide observations on magnitude are as follows.

Short-lived Prefix Hijacking on the Internet: Booth et al [3] examined data from the Oregon Route-Views project [38] for the month of December, 2005. They created a tree structure of long-term AS owners for network blocks, and then searched Route-Views for any short-lived announcements from a different origin or sub-prefix AS announced within each block. Any such short-lived announcement was suspected of being either a misconfiguration or a potential hijack as long as they occurred simultaneously. Simple mutually-exclusive changes from one AS to another were discarded as being a change to a multi-homed backup route. The final step required a manual test involving WHOIS and RADB to determine the owners of the AS blocks and filtering rules. An incident was discarded If there was a legitimate relationship. This could create false positives partially because WHOIS is often out-of-date.

The end result is that with 95% confidence they found between 26 and 95 hijacking incidents within the month. They also found 750+ misconfigurations and false alarms, many more than purposeful hijackings.

Large scale leaks: The Network Research Lab at the University of Arizona [36] examined large-scale route leaks that affected multiple network blocks. Their method was to examine suspicious announcements with respect to a network's past behavior and identifying abnormal behavior. They identified five to twenty large leaks each year that lasted from a few minutes to many hours. Figure 2.7 shows an table of large scale leaks detected in 2009.

The Arizona study had evidence that the hijacks were large in the number of networks attacked and that since they were seen by multiple vantage points, the attacks spread across a wide area of the internet. This begs the question: In addition to wide-spread hijacks, are there also attacks that



2009:

time	offender AS	AS name	AS location	# of offended ASes	# of offended prefixes	# of offended ip addresses	duration
02/14/09	8895	ISU Riyadh AS	Saudi Arabia	31	243	289,280	2.0 hours
04/07/09	36873	CELTEL	NIGERIA	15	45	27,392	10 mins
05/05/09	10834	Telefonica Data Argentina	Argentina	91	1,108	1,713,664	3.0 hours
07/12/09	29568	COMTEL	Romania	17	56	20,480	23 mins
07/22/09	8997	SPBNIT OJSC North-West Telecom	Russia	173	351	101,500,416	59 secs
08/12/09	4800	LINTASARTA	Indonesia	13	39	18,176	32 secs
08/13/09	4800	LINTASARTA	Indonesia	68	492	685,568	7.8 hours
12/04/09	31501	SPBTELEPORT	Poland	19	77	1,574,400	68 secs
12/15/09	39386	Saudi Telecom	Saudi Arabia	24	67	664,064	62 secs

Figure 2.7. *Large Scale Route Leaks Detected in 2009 (from [36])*

are highly localized? How many and how often do they occur? For example, a sub-prefix attack could be is seen in Europe, or Germany, but not spread to the rest of the world. Unfortunately I could find no study that analyzed this situation.

Network-level behavior of spammers: In 2005 Ramachandran and Feamster [41] determined that BGP hijacking is a technique used by a set of sophisticated email spammers to avoid detection. These spammers would announce a complete /8 for a period of about 10 minutes, send spam from a scattered range of IP addresses within this /8, and then withdraw the announcement (see fig 2.8). Traceroute probes sent later showed that 60% to 80% of the addresses that sent spam were found to be unreachable.

Hijacking a /8 is surprising, but has distinct advantages for hiding the identity of the spammer. A /8 allows a large range of IP addresses so that email can be sent just once from each address, and routers typically do not filter /8 announcements. The AS numbers announced were both legitimate AS numbers as well as reserved AS numbers.

In terms of quantifiable results, the authors put a bound of 1% to 10% of spam being sent with this method. This would imply that multiple 10-minute announcements were made each day, but no data is cited in this study.

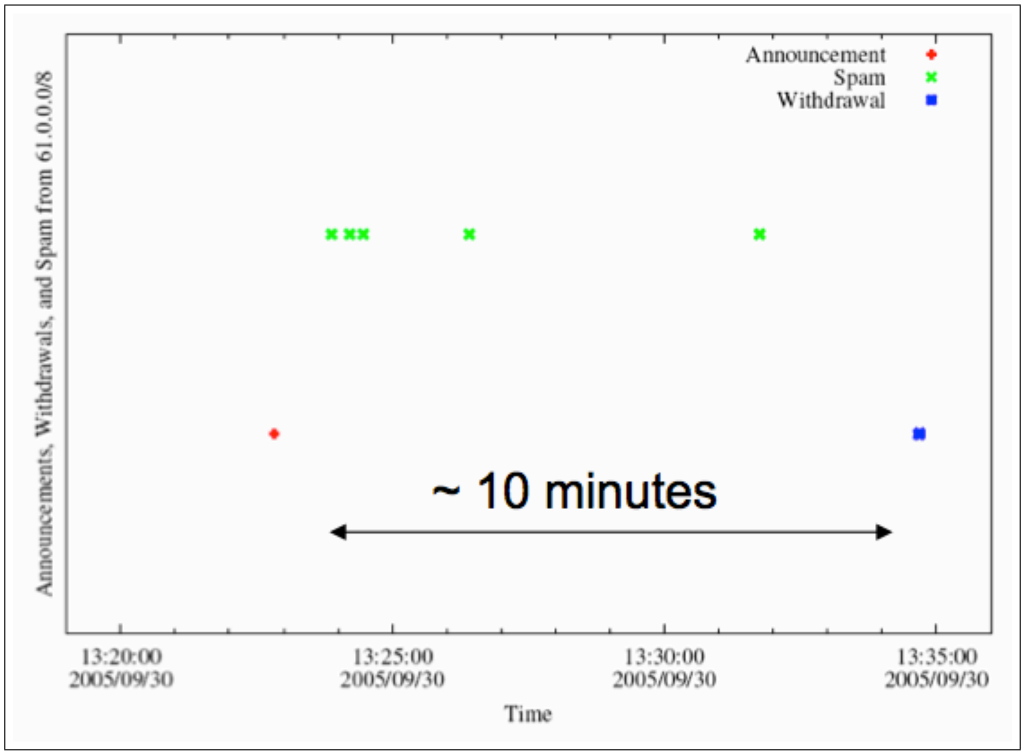


Figure 2.8. *BGP hijack from email spammers (from [41])*

Other findings: The *listen-whisper* paper [44] refers to a 2002 SIGCOMM paper on router misconfigurations. The authors claim that misconfigurations result in invalid route announcements for 200 to 1200 prefixes per day. Other papers indirectly refer to the large number of attacks that last from a few minutes to an hour or more, implying the need for real-time detection and prevention if anything realistic is hoped to be accomplished.

2.2.5. *Benevolent Attacks: Machiavellian Routing.* In contrast to the papers describing negative attack consequences, one paper describes a positive reason for an AS to poison its own route announcements. Katz [26] describes the problem when an AS failure causes long-term internet outages (more than 10 minutes, usually many hours). By using the path poisoning technique shown in Figure 2.9, the authors demonstrated how to bypass the failing AS because it will reject the route due to loop detection. The origin inflation O-O-O was a clever way to keep path lengths equal

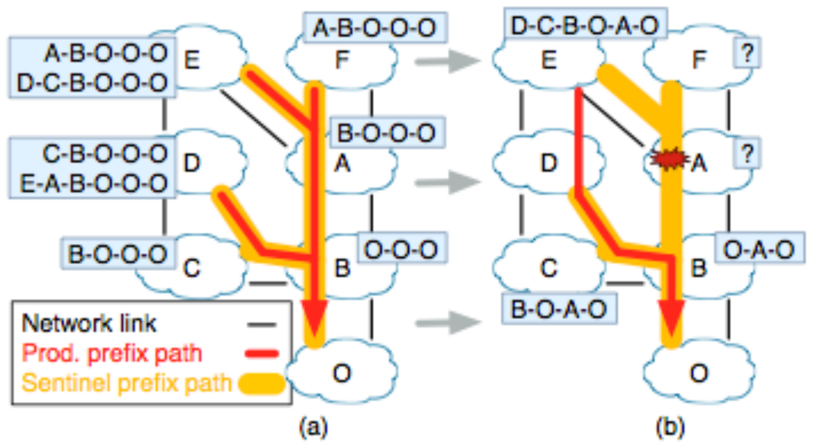


Figure 2: Routes and routing tables (a) before and (b) after *O* poisons *A* to avoid a problem. Each table shows *only paths to the production prefix*, with the in-use, most-preferred route at the top. Poisoning *A* for the production prefix causes it to withdraw its route from *E* and *F*, forcing *E* to use its less-preferred route through *D* and leaving *F* with only the sentinel. Routes to the sentinel prefix do not change, allowing *O* to check when the problem has resolved.

Figure 2.9. *Machiavellian Poisoning Example (from [26])*

during and after the removal of a self-poisoning. Their paper also referenced tools that probe the data plane (Hubble and reverse-traceroute) to determine when an AS failure occurs and how to detect that the failure has been resolved.

The Machiavellian poisoning technique was only recently proposed and needs much more study. However, It is unclear how practical and long-lived this approach will be if detection and prevention techniques nullify its core mechanism.

2.3. *Detection and Prevention Techniques*

Detection and prevention of IP hijacks requires managing the threat model. BGP routers need to verify the authenticity and completeness of the information received from others and must be able to generate authoritative information that other routers can verify. Lacking such a built-in system, various other techniques can be used to improve confidence in the correctness of routing messages, but not necessarily to 100%. This section surveys relevant research papers that describe BGP security solutions ranging from external monitoring and alerting to comprehensive built-in solutions which require infrastructure rollover.

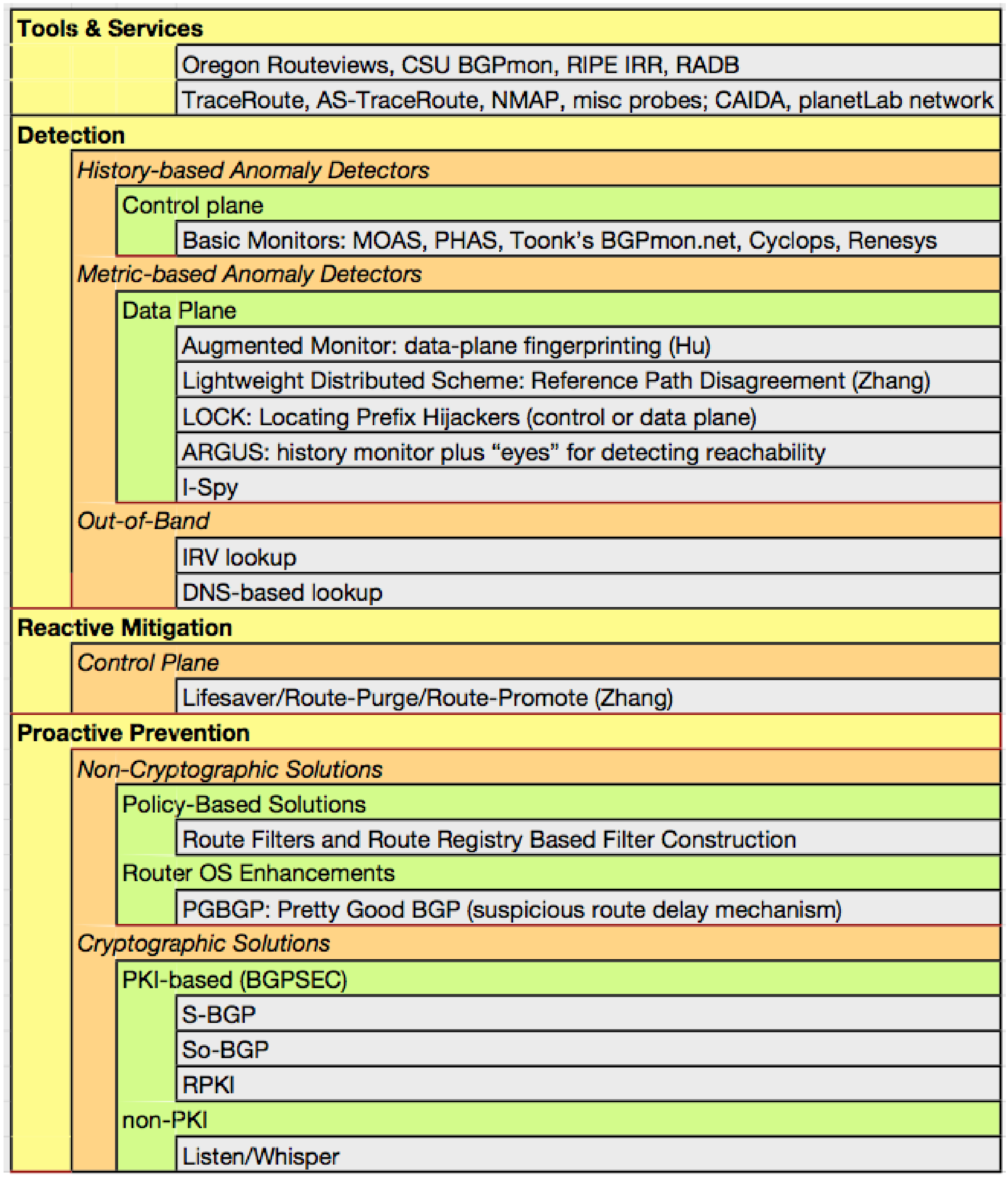
These research papers were examined with four criteria in mind: *what does the solution do* (detect, reactive mitigation, proactive prevention), *how does the solution work*, *how well does it work*, and *how difficult is it to deploy*?

Figure 2.10 exhibits a solution taxonomy organized by the first two questions, *what does it do* and *how does it work*.

The third question, *how well does it work*, is addressed using the following criteria:

- Coverage: which attacks (origin, sub-prefix, path)? does it scale?
- Accuracy: deterministic or probabilistic? does it handle evasion avoidance?
- Speed: real-time or delayed response?

Finally, there is the fourth question – *deployment cost*. Does the solution require a change to existing routers or to the BGP protocol itself? Can it be deployed incrementally or does it require a "flag day". Is it computationally expensive or require a large external infrastructure such as a PKI? Is the operational expense of deploying and maintaining the solution high or low? These questions will be discussed in-line with the solution descriptions and summarized at the end of this section. We now explore the taxonomy categories; various system names are displayed in bold font. Since this is a survey paper with a large number of systems to discuss, we present summary descriptions only, sacrificing depth for breadth.



- **Tools & Services**
  - Oregon Routeviews, CSU BGPmon, RIPE IRR, RADB
  - TraceRoute, AS-TraceRoute, NMAP, misc probes; CAIDA, planetLab network
- **Detection**
  - *History-based Anomaly Detectors*
    - Control plane
      - Basic Monitors: MOAS, PHAS, Toonk's BGPmon.net, Cyclops, Renesys
  - *Metric-based Anomaly Detectors*
    - Data Plane
      - Augmented Monitor: data-plane fingerprinting (Hu)
      - Lightweight Distributed Scheme: Reference Path Disagreement (Zhang)
      - LOCK: Locating Prefix Hijackers (control or data plane)
      - ARGUS: history monitor plus "eyes" for detecting reachability
      - I-Spy
  - *Out-of-Band*
    - IRV lookup
    - DNS-based lookup
- **Reactive Mitigation**
  - *Control Plane*
    - Lifesaver/Route-Purge/Route-Promote (Zhang)
- **Proactive Prevention**
  - *Non-Cryptographic Solutions*
    - Policy-Based Solutions
      - Route Filters and Route Registry Based Filter Construction
    - Router OS Enhancements
      - PGBGP: Pretty Good BGP (suspicious route delay mechanism)
  - *Cryptographic Solutions*
    - PKI-based (BGPSEC)
      - S-BGP
      - So-BGP
      - RPKI
    - non-PKI
      - Listen/Whisper

Figure 2.10. *Solution Taxonomy*

2.3.1. *Tools and Services.* Many of the monitoring solutions rely on historical and real-time collection of BGP announcements. The Oregon RouteViews project [38] peers with dozens of routers distributed world-wide and archives the information. Generally RouteViews data is not available real-time, there may be a delay of 15 to 30 minutes. Improvements are currently under way to provide real-time data. RIPE also maintains archives of RIBs and BGP announcements.

The CSU BGPmon project [48] builds on RouteViews and collects data from its own set of BGP peers to provide an XML data feed of BGP announcements and status messages. Individual BGPmon stations can be chained together to provide a consolidated real-time XML data feed.

The tools most often cited for probing the data plane are TraceRoute and AS-TraceRoute, an experimental tool that attempts to associate AS numbers to the IP addresses found in the trace. However, It is somewhat easy to for hijackers to avoid detection by deceiving TraceRoute via modification of the TTL in returning packets.

Other tools for determining connectivity and fingerprinting hosts included PING, NMAP, TCP/ICMP timestamp probing and IP Identifier probing. Many of the papers ran simulations of their proposals using topology information from CAIDA. Several also used the PlanetLab network cited as a testbed for running experiments.

It is interesting to note that BGP researchers aren't the only ones who use these tools. Sophisticated attackers wishing to avoid detection use TraceRoute and Oregon RouteViews to aid in cloaking their activities. Apparently research data cuts both ways: defenders can use it to assist in the detection of hijacks, and attackers can use it to make their attacks more stealthy.

2.3.2. *Detection.*

2.3.2.1. *History-Based Anomaly Detectors - BGP Monitors.* This set of solutions was inspired by a 2001 paper by Zhao *et al* regarding BGP multiple origin AS (**MOAS**) [53]. They analyzed the existence of short-lived MOAS and theorized explanations from faults, misconfigurations, and multihoming. This led to the development of external monitoring systems to detect prefix hijacks, and later, sub-prefix hijacks.

The Prefix Hijacking Alert System (**PHAS**) [28] is one such monitoring system. Its design incorporated multiple BGP data sources from RouteViews and RIPE to allow geographically diverse vantage points. The basic mechanism was for PHAS to store an "origin set" for each prefix and detect whether a new announcement added or deleted from this set. In the event of a change to the set, a notification was sent to registered users of the system. Only the end user could determine if the change was an actual hijack. PHAS also incorporated several clever features. The detection method used moving windows and exponential decay to avoid notification flapping. Notifications were sent digitally signed so they could not be spoofed. They were also set via multiple email paths in of reachability problems due to the hijack. On the receiving end, the user could set notification filters to accept or reject a notification; for example, alerts could be suppressed if the change was to a known backup AS.

In terms of the evaluation criteria, PHAS was limited in that it detected prefix hijacks only (not sub-prefix), but it scaled well, was near-real-time and simple to deploy.

Several other systems developed along similar lines. UCLA **Cyclops** [47] uses data from BGPmon and stores it in a history database. Anomalies can be detected and a user can register for notifications.

A private **BGPmon** system [46] maintained by André Toonk performs similarly with a "learning database", registration system and alerts. Other monitors mentioned were RIPE **MyASN** (discontinued) and **WatchMy.net**. A commercial monitoring system, **Renesys Routing Intelligence**, combines monitors and alerts with various analytics to keep track of internet reachability. Renesys peers with a large number of globally distributed routers and also detects sub-prefix hijacks. Their 2009 presentation at Blackhat [4] shows how their system can detect MITM attacks.

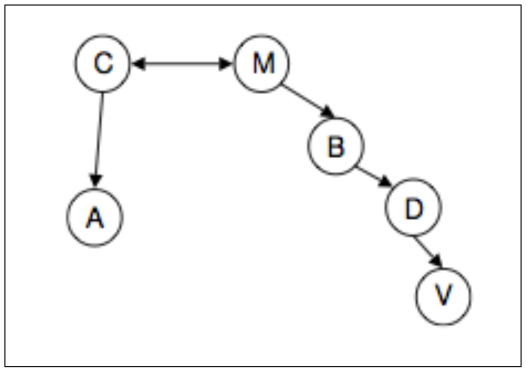


Figure 2.11. *Evasion Technique: Attacker A and Victim V announce the same prefix. Attacker A evades Detection from Monitor M due to customer route preference trumping peer route (from [44])*

Monitoring systems are only good at detecting attacks if they have wide visibility. Hijacks can be limited in scope, and the selection and distribution of monitoring stations is critical to detection. Monitors are also prone to false positives given legitimate changes that are not in the history. Furthermore, attackers can use evasion techniques to hide from monitors, as discussed in Zhang's *Practical Defense* paper [51]. An example evasion is illustrated in Figure 2.11.

2.3.2.2. *Metric-based Anomaly Detectors (Data Plane).* We now summarize five hijack detection systems based on information available in the data plane.

The first, Hu and Mao's [22] **fingerprint** system was proposed as a complementary double-check intended to reduce the number of false positives issued from BGP control-plane monitors. This was done by detecting data plane differences between an attacking AS and the legitimate owner AS.

The system uses multiple geographically dispersed stations to send probes along different paths to addresses in a suspected incident. If possible, a live host would be found at each target, probed with NMAP, timestamp and other techniques, and then analyzed for differences. Average probing time was 3 minutes, and was always less than 10 minutes. If the returning fingerprints were the same, a false positive could be avoided. If different, it strengthened the case for an actual hijack.

The authors tested their system against a variety of attack scenarios including prefix, sub-prefix, and next-hop attacks. The results were encouraging: during their tests they were able to reduce the number of origin hijacks from 10418 suspected to 137 actual attacks.

Zheng *et al* [54] created a **lightweight scheme based on hop-count** capability which is real-time, can be deployed incrementally, and runs completely in the data plane. The system periodically runs a traceRoute against registered prefixes from multiple vantage points. It records these and looks for differences in the counts over time. Some differences can be legitimate, some can indicate a hijack. To make the system more robust, they propose counting the hops from a reference router that is topologically close to the originating AS. (figure 2.12). Although not mentioned in their paper, a topologically close reference point might also help minimize evasion techniques that alter information returned from traceRoute.

The system has no dependence on live BGP monitoring. It is based on hop count stability and AS path similarity only. Alarms are issued when measurements show departure from this stability. The authors ran experiments with PlanetLab and a simulation system based on real Internet traces. The results gave high accuracy, with false positive and false negative rates below one-half percent.

This lightweight scheme has a number of desirable attributes, but may be limited in its scalability due to the necessity of carefully choosing reference points. Running 400,000 prefixes to 60,000 ASes requires too much a priori knowledge on the network topology.

The **LOCK** system proposed by Qiu *et al* [40] has somewhat different goals. It can run in either the control or data plane. But rather than *detect* a hijack, it is designed to *locate* the offending AS even if it is buried somewhere deep in a routing path.

LOCK is a system that looks for differences in geographically dispersed AS path measurements. It is based on two observations. First, hijackers cannot manipulate the portion of an AS

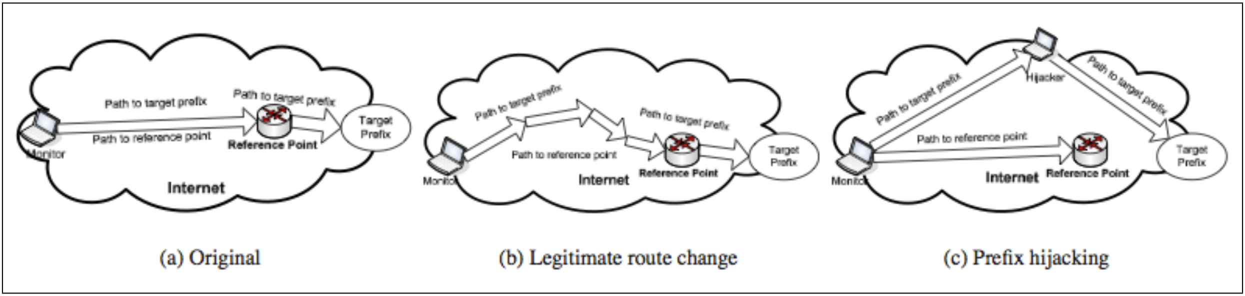


Figure 2.12. *Data Plane Detection: PathDisagreement using hop-count and a reference point (from [54])*

path from a polluted vantage point to an upstream neighbor of the hijacker. Second, the trustworthy portion of AS paths from multiple vantage points to the hijacked victim will converge around the hijacker AS. Knowing this, the LOCK monitor performs 3 steps: *clustering* of monitors that have similar paths to a target AS, *ranking* the monitors based on probability that its path is polluted, and *selecting* the highest ranked monitor in each cluster to watch the target prefix. A location algorithm then examines the differences between these monitors to determine the offending AS.

The authors tested simulations, reconstructed known hijack attempts, and conducted their own prefix attacks in PlanetLab; the results were quite accurate in locating the offending AS.

ISPY, proposed by Zhang *et al* [52], is a unique contribution in its design goals and method. ISPY is a completely self-contained and self-centric system to be run for an AS on behalf of that AS. The key idea is for ISPY to periodically perform a light-weight probe to 3000 transit ASes with traceRoute and PING, looking for cuts in connectivity. If a hijack is occurring, large swaths of the internet will not return a response to the origin AS; the response will instead be re-directed to the attacker. ISPY's detection logic differentiates between the "cut" signature for a hijack vs normal link outages. When a cut threshold larger than 10 is detected, an alert is sent to operators. This alert is guaranteed to be received because it is self-AS-originated and does not have to transit a polluted network. The system was tested with simulations and in PlanetLab, then tested the live

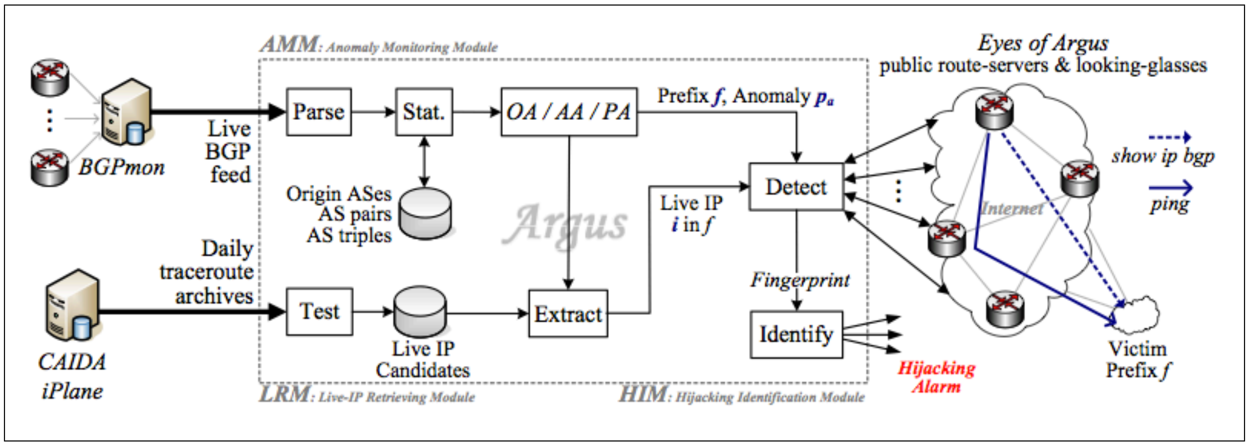


Figure 2.13. *ARGUS hijack detection architecture (from [43])*

internet; it's detection rate was extremely accurate, with low false positive/negatives. Speed was good; a probe took from approximately 1 to 3 minutes. It is easily deployable since it doesn't require vantage points but is self-contained.

ISPY appears to be a promising system, but it has limitations. It is only useful for prefix hijacks, not sub-prefix. Currently it can only detect that a hijack *is* occurring, but not *who* the attackers are. This means the system would need to be coupled with a mitigation or prevention system. Future efforts are meant to deal with these issues.

ARGUS [43] is a relatively new detector announced in 2012. ARGUS also compares BGP data to historical data as the main anomaly detection mechanism, but confirms the observation by statistical correlation on data obtained from multiple remote vantage points. ARGUS collects this data from other routers using `show ip bgp` and `ping`. Their method is based on the fact that polluted router usually can not communicate with hosts in the victim's address space, but normal routers can. The ARGUS architecture is illustrated in Figure 2.13.

2.3.2.3. *Out-of-Band Query Systems.* Huston's survey paper [24] reviews two methods for verifying route integrity based on out-of-band query systems.

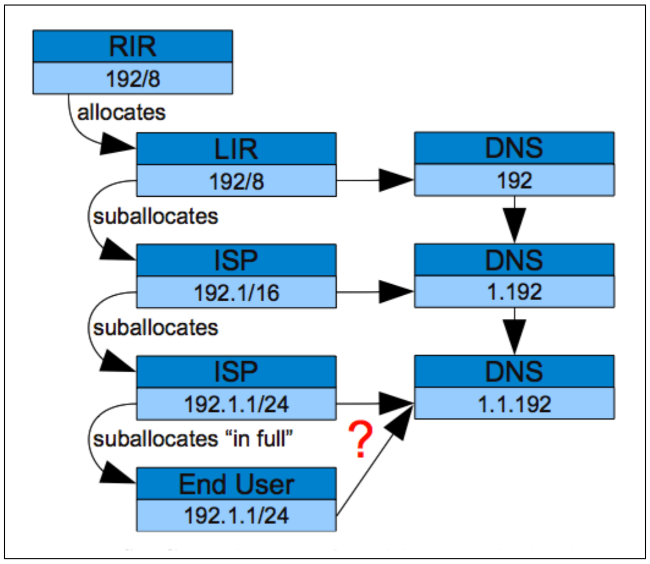


Figure 2.14. *Early BGP+DNS proposal showing problem in sub-allocation (from [24])*

IRV (Inter-domain Route Validation) is a method for distributing the information in the Internet Route Registries (IRR) into a set of per-AS route registries that can be queried out of band. Each AS would then be motivated to keep its information up-to-date. Routers would query the originating AS’s “IRV companion server” as to the authenticity of a received route. The IRV architecture had a number of issues, from finding the server to authenticating the response, and was never deployed.

The 1998 **DNS** lookup system designed by Bush and Li was also proposed as a distribution mechanism for origination information. This idea called for a new DNS zone zone, bgp.in-addr.arpa, with a new record type, the AS RR. Routers could perform a DNS query to validate origin data contained in a BGP UPDATE message.

Several issues were identified: what happens when a query isn’t answered? Is the data valid? (Data Authentication could be done with DNSSEC, but unfortunately DNSSEC was still in its infancy at this time). The approach only handled origin validation, not path validation. Huston’s critique also complains that [24] *The DNS delegation hierarchy would needed to be precisely* *aligned to the address allocation framework. Limitations to the DNS structure itself are probably the reason why the approach has never been deployed. An entire address block might be sub-allocated to a DNS sub-level (allocated "in full"), causing a prefix to refer to the DNS entry that did the allocation in full instead of the legitimate owner of the prefix. This shortcoming could only be solved with modifications to DNS.* This situation is illustrated in Figure 2.14. We consider this to be an illegitimate reason for discarding DNS, since the objective is to manage BGP security, not necessarily to maintain a structure which traces address block provenance.

2.3.3. *Reactive Mitigation.* Detection systems usually require human intervention to deal with an incident after it has been observed. Email, SNMP, and Alert Registries have all been proposed as notification systems. Unfortunately, since many attacks are short-lived, human response time may be unacceptable.

Zhang *et al* [51] propose a system in which a detection system communicates with "lifesaver" routers to execute a route **purge/promote** to eliminate as many bad paths in a network topology as possible. The detection system must be real-time, have low false-positives, and identify the victim and bogus route: the fingerprint system is satisfactory.

Bogus route purging is performed with direct communication to the lifesaver using a router with Route Control Platform modifications. The Route Promotion technique is clever - it shortens the AS path by combining many path elements into a single AS_Set, raising its preference. Figure 2.15 shows how some, but not all, ASes are affected after a bogus route purge (requires Route Control System at router) and a route promote.

The overall system is incrementally deployable and reduces polluted routes by as much as 57% with a small number of lifesaver routers, but is never 100% effective.

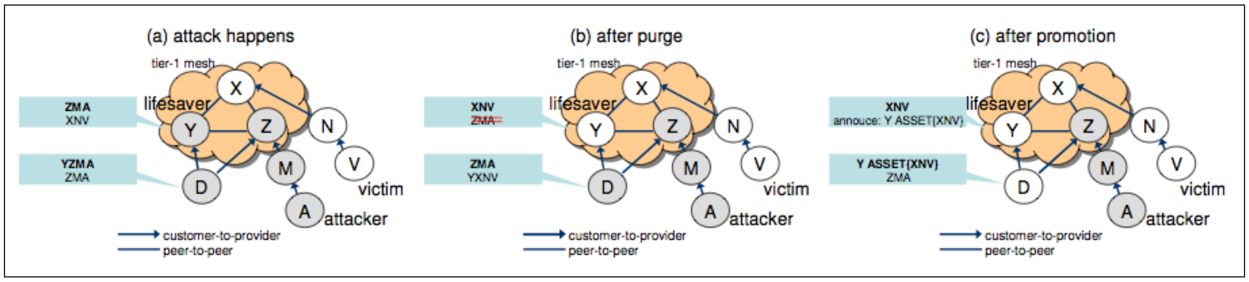


Figure 2.15. *Reactive Mitigation using Purge-Promote (from [51])*

2.3.4. *Proactive Prevention.* Rather than perform simple hijack detection or after-the-fact mitigation, various systems have been proposed to prevent routers from accepting and propagating bogus routes in the first place. Some of these systems use cryptographic hashes, signatures or PKI structures which can be computationally expensive and complex to deploy. The first systems we discuss are more simple.

2.3.4.1. *Non-Cryptographic Solutions.* The most widely deployed proactive defense is **route filters**. An AS should filter announcements from bogons and from its own customers, especially stub ASes that do not provide transit services for others. When implemented correctly filters can prevent route leaks. Unfortunately, filtering is not always done correctly; witness the Australian *Dodo* incident [45] mentioned in the introduction.

Filtering of bogus routes becomes more difficult when the erroneous information source arrives from several hops away. To assist in building better filters, Routing Registries were designed to store prefix ownership, AS connectivity and and routing policies [5]. RADB and the RIPE IRR are examples of routing registries. Information may be archived using RPSL, an XML extension for routing policy.

The problem with routing registries is that they are hard to maintain. To be effective, they must be secure, complete and accurate. Criticism of current registries is that their data can be compromised by an attacker. But the wider problem is that routing data is not kept up-to-date. ISPs merge, personnel change, and the data goes stale. Compounding this is that ISPs may reluctantly submit routing information because this may disclose local policies that the ISPs regard as confidential. Furthermore, not all ISP's participate; in fact, not even all tier-1 ISPs participate. Since the tier-1's supply access to every worldwide internet destination, this severely limits the effectiveness.

Pretty Good BGP (**PG-BGP**) described by Karlin *et al* [25] is a hijack prevention system that automatically delays the use and propagation of suspicious routes in favor of known alternatives. This delay lasts until either a human operator verifies the path or until a timeout occurs. PGBGP must be incorporated into the router's operating system via a software update.

PGBGP maintains a table of trusted routing information learned from BGP update messages. The origin isn't suspicious if it is in a previous history path. Sub-prefixes are also ok if they use the same AS as the origin.

The system uses two timers, S (*suspicious quarantine* = 24 hours) and H (*history delay* = 10 days), based on a study that shows roughly 45% of new origins and prefixes exist for less than 24 hours. A suspicious route is quarantined by setting it to the lowest LOC_PREF. The route could get selected if no better alternative existed. If the route hasn't been replaced by a new BGP UPDATE message, it is allowed to be used and propagate once the quarantine period expires. The H timer is used for startup and removal for entries in the history database. New routes not in the history database are allowed to build up over 10 days. If they do not stay in the RIB for 10 days, they are removed from the history database.

The 24 hour delay is long enough for human operators to get an alert and attempt corrective action. The authors propose an internet alert registry to handle an incident within 24 hours. Only the instigator and the victim would be alerted, since they are the most likely to be motivated to do something (and the instigator would only be motivated if it were a misconfiguration).

The results, based on an AS topology simulation, were impressive. If deployed only to 62 core ASes, PGBGP can protect 97% of ASs from malicious prefix routes and 85% from bogus sub-prefix routes. If PGBGP were deployed on all ASs, both numbers would exceed 99%.

In evaluating PGBGP, it is incrementally deployable with immediate benefit, but is only effective if the core adopts it. It does not require any protocol changes. Its deployment cost is high, however, because it requires changing the router OS. Although it detects anomalies in real-time, the 24 hour delay may be unacceptable for ASes undergoing traffic management or backup route switches.

2.3.4.2. *Cryptographic Solutions.* A comprehensive security model for BGP would verify BGP messages for authenticity, integrity and authorization – that they come from a verifiable owner with a verifiably-owned address block. According to Huston [24] *One method of preventing bogus routes is to register all ASs and their prefixes so that routing announcements can be cryptographically verified.*

Cryptographic techniques can use a public key infrastructure (PKI) to establish a chain-of-trust to ensure message authenticity and integrity. This section summarizes several of these designs plus an alternative design that avoids the use of PKI entirely.

SBGP is the most comprehensive security architecture proposed. SBGP uses a PKI to digitally sign route updates as they traverse the network, chaining signatures together for path validation (see Figure 2.18). Unfortunately SBGP changes everything from the protocol and policies and is computationally expensive. These issues result in extremely large barriers to SBGP deployment. And to be effective, it would require full, or at least very wide implementation across the Internet.

SoBGP (Secure Origin BGP) is a little simpler and relaxes the constraint on path verification. It provides trade-offs regarding security and protocol overhead, combining proactive security with anomaly detection.

RPKI (Resource Public Key Infrastructure) is the current proposal with the largest momentum [23]. RPKI is in active development at the IETF SIDR working group (Secure Interdomain Routing). It uses ideas from S*BGP, but currently limits itself to origin validation. The larger BGPSec effort at the IETF follows after RPKI to secure path validation.

RPKI establishes a root of trust and uses a chain of X.509 cryptographically signed certificates to attest the "right to use" of resources such as an AS number and IP address blocks. Each certificate in the chain can sub-allocate a resource and attest to its authorization. This allows 3rd party verification. The X.509 chain is shown in figure 2.16, illustrating the delegations from IANA to the Regional Internet Registries, to Local Internet Registries and on down.

The PKI structure builds this trust chain, but has complexity issues with certificate distribution and revocation. More importantly there is the open question on how to deal with legacy IP address space. Many holders of legacy /8's are not likely to enter into an agreement with IANA giving them authority over their address space. Therefore there are likely to be large holes in the certificate chain.

The next major element of RPKI is the Route Origin Authorization (ROA). A ROA is a digital object signed by a certificate that contains an AS number and a list of authorized IP address blocks. It attests that AS X is authorized to originate any of the prefixes in the list. ROAs are published in the RPKI repository and distributed to routers via an as yet undetermined mechanism. Once there, the ROA can be used to proactively filter a prefix announcement: it will allow a prefix announcement if the origin matches as valid, deny an announcement (by setting LOCAL_PREF

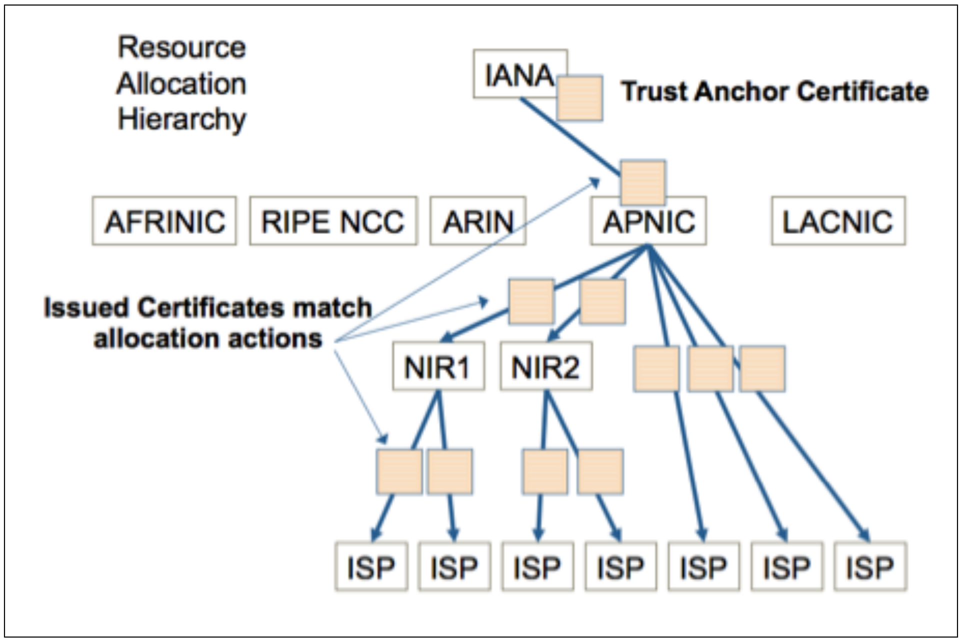


Figure 2.16. *RPKI certificate hierarchy (from [23])*

very low) if the origin is bogus, or do nothing if there is no matching ROA. This is illustrated in Figure 2.17.

The good news is that compared to S*BGP, RPKI can be incrementally deployed. On the other hand, there is much debate over the practicality and barriers to deployment for RPKI. There is already a mistrust of PKIs in general. Router code has to be updated or new infrastructure rolled out. Already complicated router policy has to change. Creation of the first batch of signed ROAs had hundreds of errors, which would cause operational errors [49]. These are all the subjects of test labs at RIPE, NIST and other places. Even if some elements can be simplified, the ISP operator community may be reluctant to deploy a system as complex as RPKI.

Cryptographic solutions offer strong authentication, but as mentioned, there is much criticism and reluctance to using a PKI. An interesting alternative is the system proposed by Subramanian *et al* [44] called **LISTEN/WHISPER**. This is a two-part solution: *Listen* checks the data plane

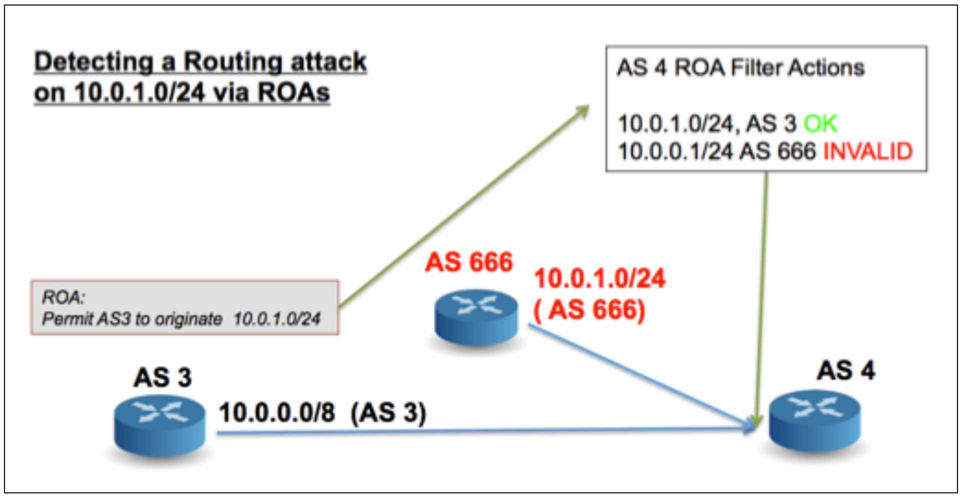

Detecting a Routing attack on 10.0.1.0/24 via ROAs
AS 4 ROA Filter Actions
10.0.1.0/24, AS 3 OK
10.0.0.1/24 AS 666 INVALID
ROA: Permit AS3 to originate 10.0.1.0/24
AS 666
10.0.1.0/24 (AS 666)
AS 3
10.0.0.0/8 (AS 3)
AS 4

Figure 2.17. *Detecting an origin hijack using a ROA (from [23])*

to determine whether destination routes are working. *Whisper* uses cryptographic functions in the control plane to detect bogus routes.

Whisper does NOT rely on a PKI. Instead it uses RSA signatures that are attached to a BGP update message in the community string. This eliminates the need for a BGP protocol change, but does require router software modification. Each succeeding router adds its signature to the chain. But without a PKI, how do we check the signature? The solution is to compare the signatures from two distinct paths, if they are different, one of them must be bogus (see Figure 2.18). If they are the same, either both are good, or both are bogus. Bogus routes were assigned penalties in the routing policy so they would be less likely to be chosen. Note that this system requires ubiquitous deployment to detect inconsistent routes.

Listen complements *whisper* by passively checking the data plane using libpcap. This is different from prior data-plane solutions which do active probing. In its simplest form, *listen* determines that a TCP flow is complete if it observes a SYN packet followed by a DATA packet, otherwise it is incomplete if no response is heard within 2 minutes. Much more has to be done to deal with false negatives and colluding attackers; in fact, smart adversaries can defeat LISTEN.

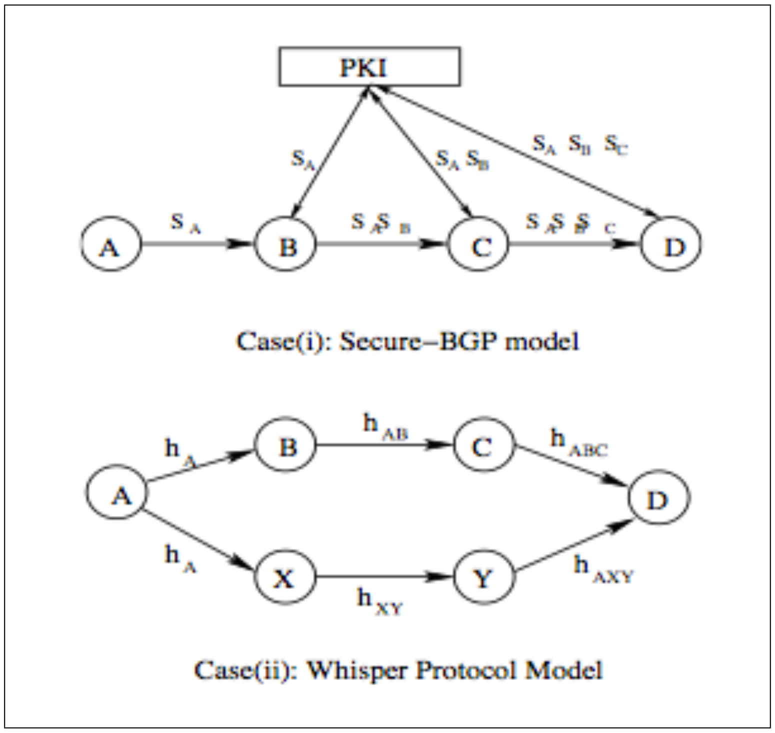


Figure 2.18. *S-BGP vs Whisper (from [44])*

The combination of *listen/whisper* gives very good results, given its cryptographic nature and reachability double-check, but it is still non-deterministic. Additionally, the deployment costs are much too high for the incremental benefit gained.

2.4. *Factors Affecting Solution Deployment*

All of the solutions surveyed have strengths and weaknesses. Despite the range of approaches presented, we argue that none of the surveyed systems is likely to see widespread deployment. This concern has also been raised by Butler, Rexford *et al* [5]. The main blockers to deployment are largely cost and complexity.

No one disputes that IP hijacking can have serious consequences. The controversy is in whether the cost of implementing any of the proposed solutions is worth the effort.

Importance to any particular stakeholder is a relative term. To characterize stakeholder importance we use a standard formula from IT risk assessment:

$$cost < probability * magnitude\_of\_loss.$$

That is, the expected loss from "doing nothing" must exceed the cost to "buy an insurance policy" or there is no motivation to act. If the loss is low or the probability is low, the issue won't be important to the stakeholder.

If all things were statistically equal, the probability of experiencing a hijack is quite low. A recent paper describing the *Argus* detection system [43] states that over a period of one year the authors observed 220 stable hijackings among 40,000 BGP anomalies. It took less than two minutes for some of these attacks to spread across 90% of the internet. 20% of these lasted less than 10 minutes. These numbers are reasonably close to those in the 2006 paper by Booth *et al* [3] that measured 26 to 95 hijackings in one month.

Using these numbers, we can calculate the probability of a hijack among one of 60,000 autonomous systems as being `220/60000 = 0.36% per year`, assuming equal distribution of attack targets. Any single ISP is unlikely to be motivated by such a low number. (Note, however, some targets are MUCH more attractive than others, raising a particular organization's probability significantly). `Probability` is only one factor in the equation, however. `Loss` is the other factor. And, as we shall see, the loss potential among various stakeholders varies dramatically.

To illustrate, we show the broad range of risk exposure borne by various ISPs. At the highest end, we have the October 16, 2012 issue of CNET quoting U.S. Defense Secretary Leon Panetta, *"A cyber-attack perpetrated by nation states or violent extremist groups could be as destructive as the terrorist attack on 9/11. Such a destructive cyber-terrorist attack could virtually paralyze the nation. They could use these kinds of cyber tools to gain control of critical switches. They could,* *for example, derail passenger trains or even more dangerous, derail trains loaded with lethal chemicals. They could contaminate the water supply in major cities or shutdown the power grid across large parts of the country. The most destructive scenarios involve cyber actors launching several attacks on our critical infrastructure at one time, in combination with a physical attack on our country. Attackers could also seek to disable or degrade critical military systems and communication networks. The collective result of these kinds of attacks could be a cyber-Pearl Harbor.*"

Also on the high-risk side are major corporations hosting e-commerce sites. Even a brief outage due to a hijack could result in a loss ranging from thousands to millions of dollars. As stated in the U.S. Department of Commerce 2013 second quarter report [37], "*The Census Bureau of the Department of Commerce announced today that the estimate of U.S. retail e-commerce sales for the second quarter of 2013, adjusted for seasonal variation, but not for price changes, was $64.8 billion, an increase of 4.9 percent (0.9%) from the first quarter of 2013.*" This equates to $700 million per day, or half a million dollars per minute, for all online transactions spread evenly over the time period.

Transactions are not spread evenly, however. There are volume differences between large and small e-commerce sites and there are also large swings in seasonality, especially during the Christmas holiday season.

To illustrate seasonality: According to the U.S. Department of Commerce, total online revenue for 2012 was $225.5 billion. The web measurement firm comScore states a lower figure, $186.2 billion. If revenues were spread evenly across the year, each day would result in $510 million online revenue. In contrast, comScore states that online revenues for the holiday shopping period were $56.78 billion, or 30% of the total revenue. Green Monday (the Monday 10 days before

Table 2.1. *E-commerce statistics for 2012 sourced from comScore, annual reports and private communications*

	Annual Revenue	Average Day $ per minute	High Sales Day $ per minute
U.S. Total online sales	186,200 MM$	326 K$	787 K$
Amazon	61,090 MM$	106 K$	258 K$
Large Computer Corp	2,250 MM$	3.9 K$	9.5 K$

Christmas, otherwise known as the last chance to buy before it is too late) is the largest volume day with $1.133 billion in online revenue. This is 2.2 times larger than the evenly spread rate, and is 0.6% of the total online revenue. Assuming the holiday season is 90 days, the other 275 days have a lower average rate of $470 million per day.

Table 2.1 presents a summary of these figures and also estimates online sales rates for Amazon and a large computer manufacturer. Other large online retailers not illustrated include Google, Apple, Wal-Mart, Target, Sears, Best Buy and Macy's.

At the low-risk end, we have the very large number of small and medium ISPs that simply do not expect much loss if they should experience a hijack. For them, it appears that the problem doesn't really have an impact unless they have a critical customer willing to pay for a solution. They have little motivation or profit incentive to deploy a solution on their own. This lack of urgency at the low end drags down the deployment effort needed by the highly impacted organizations.

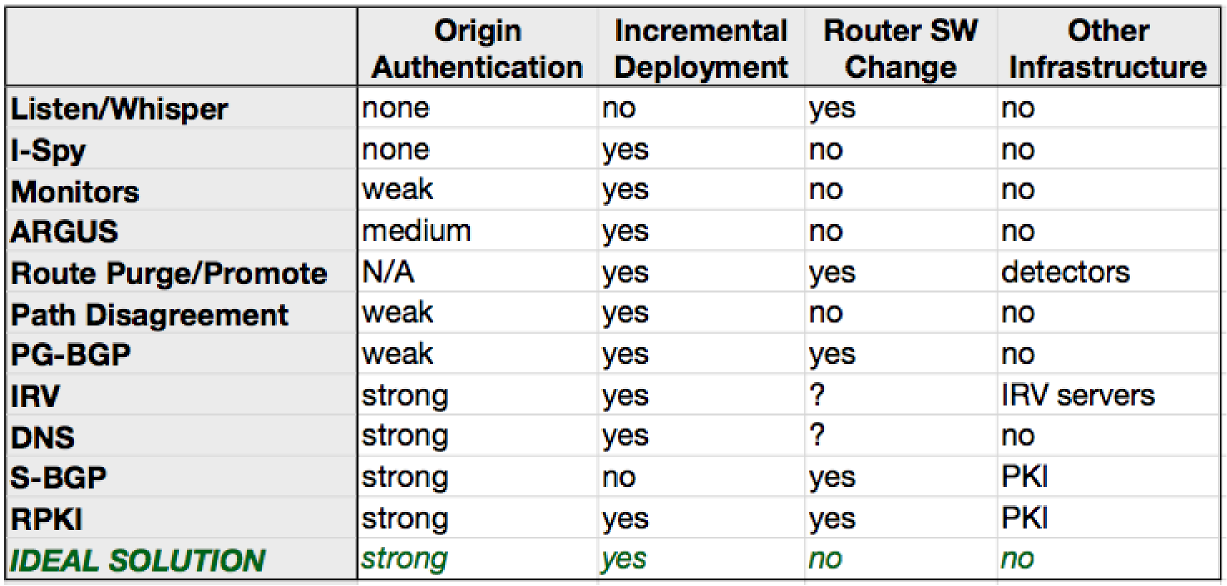
To make matters worse, if a highly impacted retailer is the subject of an IP hijack, it is quite possible that thousands of ASes, both large and small, will have a bogus route to the retailer. These thousands of ASes will be unable to reach the retailer's web site. The important fact is that you must not just protect the retailer's AS. It is these thousands of other ASes that need protection because they host the bulk of the buying customers.

In other words, there is a strong inter-dependence between ASes. Just because an ISP doesn't host a ”critical retailer” doesn't mean that they don't host ”critical customers for other retailers”. This inter-dependence requires an IP hijacking solution that prevents bogus routes at both large and small ASes. Many of the proposed detection and prevention techniques require a critical mass of ISPs to participate in the solution. This inter-dependence therefore sets the stage for several deployment objectives:

- the solution must not require the participation of everyone, nor require a “flag-day” event. It must be able to be incrementally deployed.
- a proposed solution must be cost-effective for all parties involved. It must not require replacement of existing infrastructure (new routers), nor expensive investment in new infrastructure.
- the solution must be easy to understand and deploy. Complexity is the enemy.
- the solution should fail-safe. In fact, risk from human error must be minimized. If an ISP makes a mistake, it must not jeopardize the ISP or impact the remainder of the internet.
- the solution must have strong detection capabilities, minimizing false positives and negatives. Prevention mechanisms must act in a timely manner.

Any solution that does not meet these needs is unlikely to be accepted or deployed. They will be simply ignored.

In general, most of these solutions surveyed do not meet these objectives. They are either incomplete, too weak, too complex, too costly, or do not allow incremental deployment. The level of AS participation required for the solution to be effective varies among the solutions. Anomaly detectors require no participation. Mitigation and prevention solutions vary: some require critical mass at the core, some require total participation.



	Origin Authentication	Incremental Deployment	Router SW Change	Other Infrastructure
Listen/Whisper	none	no	yes	no
I-Spy	none	yes	no	no
Monitors	weak	yes	no	no
ARGUS	medium	yes	no	no
Route Purge/Promote	N/A	yes	yes	detectors
Path Disagreement	weak	yes	no	no
PG-BGP	weak	yes	yes	no
IRV	strong	yes	?	IRV servers
DNS	strong	yes	?	no
S-BGP	strong	no	yes	PKI
RPKI	strong	yes	yes	PKI
IDEAL SOLUTION	*strong*	*yes*	*no*	*no*

Figure 2.19. *Proposed Solutions evaluated against Deployment Factors*

These attributes are summarized in Figure 2.19. The *ideal solution* must have strong authentication for prefix origins so that its detection mechanism avoids false positive and negative alarms. The solution must be able to be incrementally deployed, and must be cost-effective, requiring no router changes (other than some minor configuration), or new infrastructure deployment. Other factors include simplicity and ease-of-use. Operators will not deploy something they do not understand or fear will cause more problems than it solves. Finally, there is the issue of critical mass. The internet does not have a governing body which can mandate participation in a BGP hijacking solution. Simply put, a large majority of ASes will see no benefit in spending money on a problem that barely affects them.

Given this argument, one can examine the solutions in Figure 2.19 and make the following conclusions:

- First, DETECTORS are low-impact; an accurate, real-time system such as ARGUS may be used by interested parties

- Second, The only likely survivor from the mitigation/prevention category is RPKI. All other solutions fail to meet the objectives and BGPSEC is too invasive.
- Third, even RPKI is unlikely to be widely deployed due to its deployment costs, data distribution method and router upgrades. PKIs are complex, have had trust issues with hacking and multiple trust anchors, and revocation lists are difficult to manage.
- Fourth, these shortcomings leave room for new proposals that better match the design objectives and deployment costs. This dissertation proposes ROVER to fill that role.
- Finally, it is naive to believe that a “better mousetrap” will be naturally accepted. There is much momentum and investment behind RPKI. It may be too late for a new proposal, or too complex for two solutions (RPKI and ROVER) to mutually co-exist, handling overlaps and possibly contradictory data declarations.

In the end, it may be that one, two, or none, of these solutions is deployed. If the answer is “do nothing”, then much more serious work will need to be done to bolster the current approach to IP hijacks – find a method to do better prefix filtering. Unfortunately better filtering cannot really be done without a secure source of authoritative route origin data. To meet this need, as well as to provide mechanisms for hijack detection, this dissertation proposes ROVER as a solution that overcomes the previously listed barriers to adoption.

CHAPTER 3

ROVER

Stripped to its essentials, ROVER is simply a method for publishing and retrieving route origin data via the publicly available reverse-DNS infrastructure. These DNS data are protected from fraud by requiring the use of the DNSSEC extension to DNS.

The ROVER mechanisms by themselves do not directly perform IP hijack detection and prevention. Instead, ROVER is an enabling technology. The ROVER mechanisms are meant to be embedded into other fully functional applications. For example, ROVER can be incorporated into a variety of operational models including fully dynamic in-line BGP filtering, periodically updated authenticated route filters, and a hijack detection system that issues real-time notifications for network operators.

This chapter explains the ROVER design principles and architecture. It then describes the ROVER components: the DNS naming convention, resource records for publishing ROVER data, and the ROVER validation algorithm. The chapter ends with a description of the live Internet testbed as an example of a ROVER application.

3.1. *ROVER Design Principles*

The ROVER design was deeply influenced by the fact that other solutions to the IP hijacking problem were not being widely deployed. Because of this, we developed a list of key design principles based on the *solution deployment objectives* outlined in Section 2.4 (e.g. *allow incremental deployment, be cost-effective* and *fail-safe*). The ROVER design principles are:

- Use DNSSEC signed reverse-DNS as an out-of-band advisory mechanism to store and retrieve authoritative BGP origin data.

- Use the existing cryptographic root-of-trust and infrastructure since it has been deployed and has solved the political issues associated with internet governance.
- Assure the system will not break what is working today and must fail-safe. When disabled, routing will work just as it does today without ROVER. When enabled, ROVER protection enhances BGP security.
- Any viable solution must be publicly checkable. Anyone must be able to ask "which AS originates this address block"?
- The solution must be cost-effective and align operational costs with benefits. There should be incentives for keeping origin data published and up-to-date.
- Manage operational issues with IP address ownership and assignment
  - IP Address owners must be able to maintain their own authorization information
  - An IP address owner must be able to directly authorize origin, transit and other BGP route security information.
  - If an owner assigns an IP address block to a 3rd party, they an act either as an agent, or delegate publishing authority to that 3rd party.

ROVER relies on *existing* DNS and *no changes* are made to the DNS protocol. End applications that perform verification must understand the naming convention and the resource record types. Verification also requires support for DNSSEC. Again, no changes in the DNSSEC protocol or behavior are introduced.

3.2. *ROVER Design Model*

Figure 3.1 illustrates the ROVER design model. The base of the hourglass depicts the *re-use* and *leverage* of a large set of deployed, well-understood, and well-tested infrastructure components centered around the reverse-DNS itself. Using existing infrastructure lowers the entry barriers in

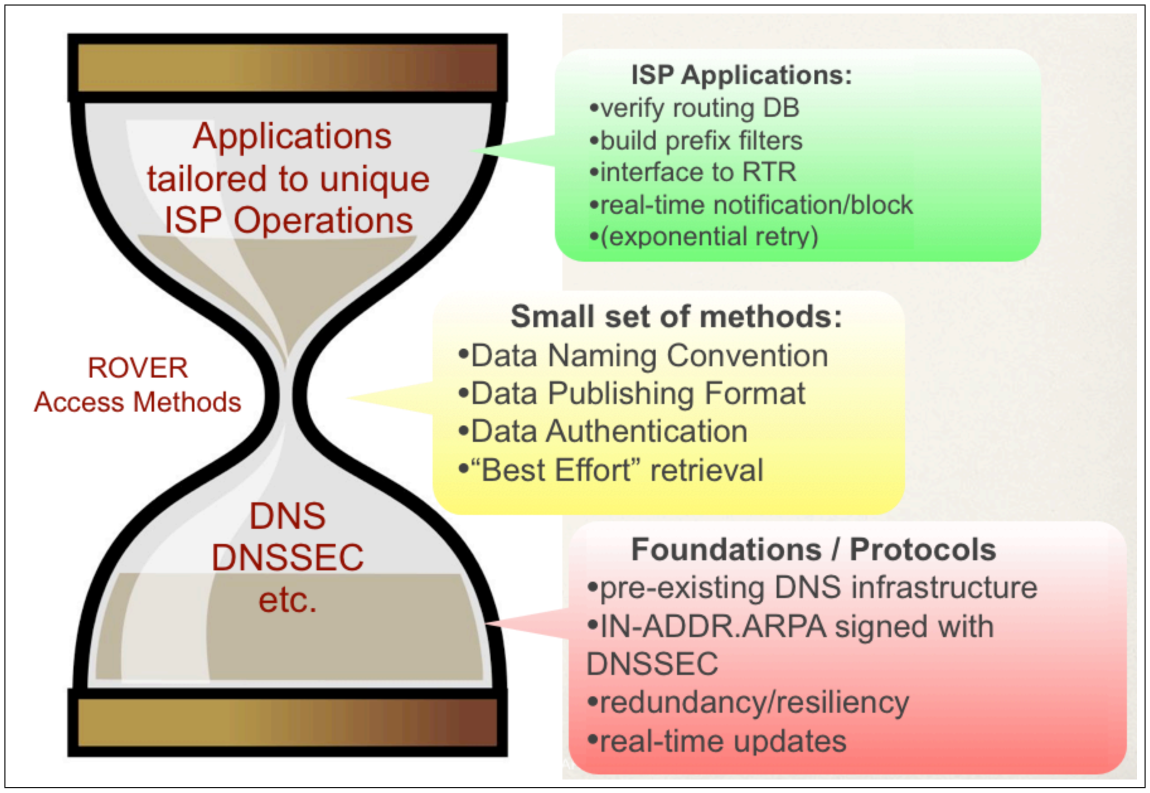


Figure 3.1. *ROVER Design Model*

terms of cost, ease-of-use, and understanding. This infrastructure also lends itself to rapid dissemination of data. An ISP can add or delete origin information via DNS updates and they will be available world-wide within minutes, as opposed to other techniques such as PKI distribution methods and certificate revocation. It is important to note that ROVER makes no changes to this base infrastructure. In particular, ROVER makes no protocol changes and requires no modifications to any DNS server, cache, or resolver implementation.

The key to the design lies in the small neck of the hourglass depicted in Figure 3.1. Only a small set of ROVER methods are common to all applications. These include:

- A DNS naming convention for CIDR address blocks. This naming convention covers both legacy address space and the address blocks assigned from IANA's Regional Internet Registries.

- A set of DNS resource records for publishing BGP route data
- DNSSEC authentication using an internationally agreed root-of-trust signing key. (RPKI has not settled on a single root of trust at this time).
- "Best Effort" retrieval. Any failure to retrieve data must be handled at the higher level application layer.

Finally, the large top of the hourglass depicts the broad range of ROVER-based applications. This system is intentionally designed to allow ISP and other organizations to run their shops in their own unique preferred operational manner and not dictate a single solution or operational process. Some ISPs will prefer to use IP hijack detection and alerts. Some will want to deploy a preventive mechanism. Still others may want to verify IRR and RADB information to build route filter lists. Some ISPs may do nothing more than publish origin data in the reverse-DNS and let other ISPs deal with hijack incidents.

3.3. *ROVER Publishing*

ROVER publishes data in the DNS by introducing two new components; a new naming convention that uniquely names a prefix and new DNS resource records used to store routing data. For example, ROVER provides a unique DNS name for the prefix 129.82.128.0/18 and stores a DNS resource record at that name which authorizes ASN 12145 to announce the prefix.

3.3.1. *The ROVER Naming Convention.*

The reverse DNS has a standard naming convention defined for both IPv4 and IPv6 addresses. The most common use of these names is to associate an IP address with a host name by means of a PTR resource record. As an example, IP address 129.82.138.2 is encoded as domain name `2.138.82.129.in-addr.arpa` (note the simple reversal of octets in the domain name). A PTR record retrieved from that DNS domain name identifies the host as `alpha.netsec.colostate.edu`.

The reverse DNS naming convention is designed to handle a single address at a time. ROVER, however, needs a method that can specify a *block* of addresses, not just a single host address. Address blocks are written in CIDR "/" notation. The prefix length can be at at any arbitrary network boundary (for example: 129.82.128/17). The new naming convention must be able to encode this.

To accomplish this, the ROVER naming convention defined in *Algorithm 1* constructs DNS names that take advantage of the hierarchical tree structure of both the Reverse DNS and the IP address structure.

- In the DNS naming hierarchy, `128.82.129.in-addr.arpa` is logically below `82.129.in-addr.arpa`, which is logically below `129.in-addr.arpa`. Other flat approaches to naming, such as Distributed Hash Tables, have been proposed, but the DNS tree structure remains a powerful abstraction. It forms the basis for the operation of DNS; caching, delegation, DNSSEC signing, and so forth all benefit from the DNS tree structure.
- Similar to the DNS hierarchy, IP addresses also have a logical tree structure where 129.82.128.0/24 is subprefix (logically below) 129.82.0.0/16 which is a subprefix of 129.0.0.0/8.

This alignment between the DNS hierarchy and the IP address hierarchy serves both systems well. It also allows one to easily encode prefixes that fall on an octet boundary (e.g. IPv4 prefixes whose mask length is a multiple of 8). The challenge is to preserve this alignment even when CIDR prefixes do not fall on octet boundaries. For example, 129.82.128.0/19 is a subprefix of

Algorithm 1 Pseduocode to convert an IP prefix in CIDR notation to a reverse DNS name. This is for IPv4. IPv6 is similar but operates at nibble instead of octet boundaries.

```
function CONVERTTOREVERSEDNS( $IPprefix$)
    $octets \leftarrow IPprefix_{address}$                          // Split IP prefix into address and length
    $prefixLength \leftarrow IPprefix_{prefix}$
    $bitCount \leftarrow prefixLength \bmod 8$                                  // number of leftover bits
    $reversedOctets \leftarrow reverse(octets)$                                    // Reverse the octets
    if $bitCount = 0$ then                                                       // at octet boundary?
        return 'm.' + $reversedOctets$ + '.in-addr.arpa'                         // at octet boundary
    else                                                                     // not at octet boundary
        $firstOctet, lastOctets \leftarrow splitOnce(reversedOctets, '.')$               // peel off first octet
        $DNSname \leftarrow$ 'm.' + $lastOctets$ + '.in-addr.arpa'
        $bitMask = [128, 64, 32, 16, 8, 4, 2, 1]$
        $index \leftarrow 0$
        while $index \neq bitCount$ do                    // Add in the binary subdomains for leftover bits
            if $firstOctet \cdot bitMask[index]$ then
                $DNSname \leftarrow$ '1.' + $DNSname$
            else
                $DNSname \leftarrow$ '0.' + $DNSname$
            $index \leftarrow index + 1$
        return $DNSname$
```

129.82.128.0/18. The DNS name for 129.82.128.0/19 should be logically below the DNS name for 129.82.128.0/18.

Previous attempts at constructing DNS names for prefixes have not captured this relationship. Most notably, [2] proposed to encode the prefix 10.1.128/20 as the DNS name `128/20.1.10.bgp.in-addr.arpa`. One important limitation of this approach is that it does not reflect the IP address allocation hierarchy. To see this, consider the prefixes 10.1.128/20 and 10.1.128/21. In the IP hierarchy, prefix 10.1.128/20 is the parent of 10.1.128/21. This relationship is not captured in the DNS hierarchy. Instead, the DNS names are encoded as siblings and this creates a host of management problems. Specifically, the 10.1.128/21 is encoded as `128/21.1.10.bgp.in-addr.arpa` and prefix 10.1.128/20 is encoded as `128/20.1.10.bgp.in-addr.arpa`. When viewed as DNS names, they are siblings.

IP Prefix	Domain Name
`129.82.0.0/16`	`m.82.129.in-addr.arpa`
`129.82.0.0/17`	`0.m.82.129.in-addr.arpa`
`129.82.128.0/17`	`1.m.82.129.in-addr.arpa`
`129.82.0.0/18`	`0.0.m.82.129.in-addr.arpa`
`129.82.64.0/18`	`1.0.m.82.129.in-addr.arpa`
`129.82.128.0/18`	`0.1.m.82.129.in-addr.arpa`
`129.82.192.0/18`	`1.1.m.82.129.in-addr.arpa`
`129.82.192.0/23`	`0.0.0.0.0.1.1.m.82.129.in-addr.arpa`
`129.82.138.0/24`	`m.138.82.129.in-addr.arpa`
`129.82.138.0/25`	`0.m.138.82.129.in-addr.arpa`

Table 3.1. *IP prefixes mapped to Reverse-DNS domain names*

In contrast, the ROVER naming scheme captures the fact that 10.1.128/20 is the parent of 10.1.128/21. Since the DNS and IP hierarchies aligned, one could delegate 10.1.128/20 to an organization and that organization could then create records for sub-prefixes such as 10.1.128/21 and could even further delegate that prefix block to yet another organization.

Delegation is achieved by using existing reverse DNS names for the first part of the prefix. The ROVER naming convention then then switches to binary notation for the last $mask \bmod 8$ bits of the prefix. As can be seen in table 3.1, the result looks like an ENUM or IPv6 reverse-DNS address; that is, a string of chained sub-names. This name-chaining creates the desired effect of enabling a DNS zone delegation at any point in the chain. For example, the naming scheme allows for the creation of two /17s from a /16, two /18s from a /17.

There is one exception to the simple delegation scheme. This exception is necessary for backwards compatibility with the existing `in-addr.arpa` structure. In the IP hierarchy, $129.0.0.0/15$ is the parent of $129.0.0.0/16$. The existing reverse DNS has already established $129.0.0.0/8$ as the parent of $129.0.0.0/16$. The naming convention cannot change this existing structure since it is already widely deployed. When crossing octet boundaries the naming convention simply requires placing the delegation at the existing zone. The IP and DNS hierarchies align at all other names.

Since the ROVER naming scheme is only a convention it requires no changes to the DNS servers, caches, or resolvers. It simply provides a way to express a prefix as a unique DNS name. DNS zone administrators can choose to associate any name with a prefix, but having a common convention facilities inter-operability between organizations and allows ROVER verifiers to easily find and authenticate routing data published by other organizations.

In summary, the ROVER naming convention is designed to satisfy the following properties:

(1) *Unambiguous:* Every possible IP prefix must have a unique name and a name must uniquely match a single prefix. If there were multiple DNS names for the same prefix, applications might need to query data at each of the multiple names. Worse still, the different names could contain conflicting information. To avoid this, we require each prefix have exactly one unique DNS name.

(2) *Autonomy:* The owner of a reverse-DNS zone file associated with a CIDR address block should be able to act independently from any other organization when creating or modifying data records within the DNS zone.

(3) *Coverage Authority:* With the exception of data that has been sub-delegated to a child zone, the reverse DNS zone must be authoritative for all sub-prefixes below the covering prefix. Any query for a sub-prefix must be answered with a data record or DNS NXDOMAIN response specifying the zone as the authority.

(4) *Delegation:* It must allow the zone owner to delegate smaller address blocks to a child zone which will be independently managed.

(5) *Conformance:* It should align with naming conventions and delegation structures already in use by the RIRs for `in-addr.arpa` and `ip6.arpa`.

(6) *Simplicity:* The naming structure should be understandable, or at a minimum, able to be easily constructed by software provisioning tools and utilities such as `DIG`.

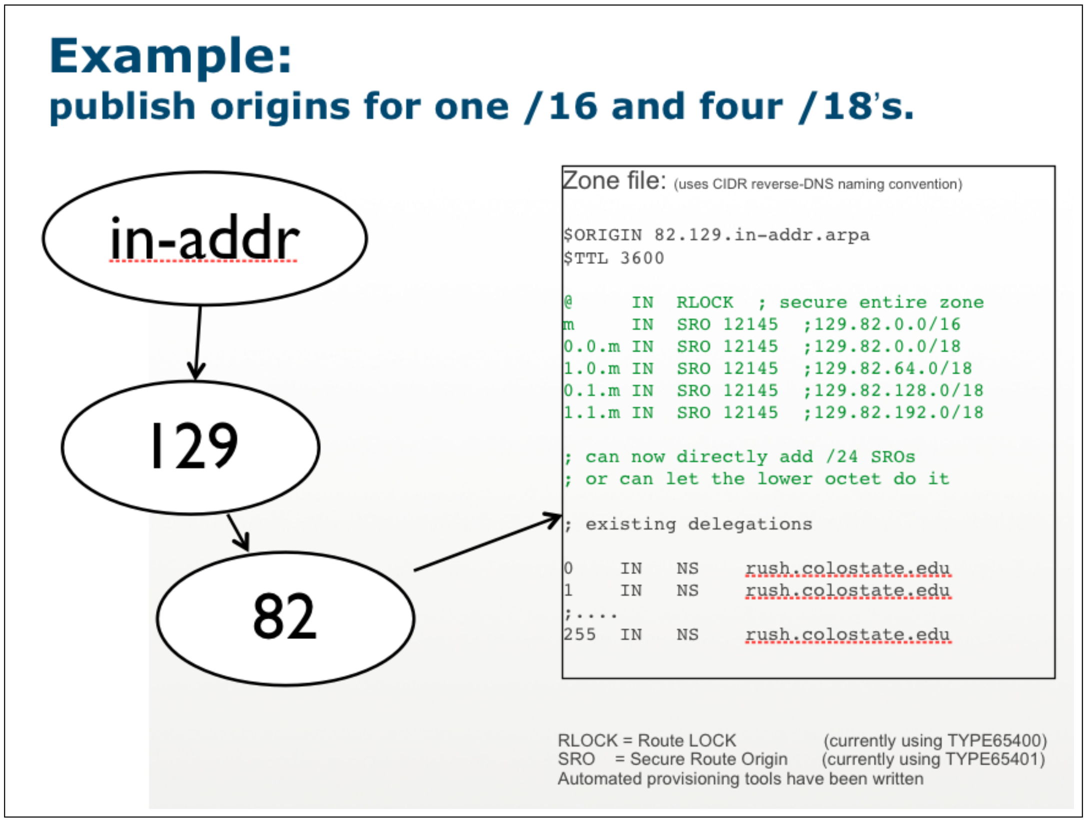
[10] provides more discussion of the naming convention requirements and provides a more detailed discussion of both the reverse DNS and previous attempts at naming prefixes. [14] provides a more comprehensive explanation of the naming convention and also shows how the same convention works for IPv6 prefixes.

3.3.2. *ROVER Resource Records.* Having associated a unique name with a prefix, one can now store data at the prefix and sign the data using DNSSEC.

ROVER introduces a new Secure Route Origin (SRO) resource records to identify authorized origin Autonomous System Number(s) for a prefix. The SRO record contains one mandatory field, the *Origin ASN* for the prefix. The *Origin ASN* field specifies an AS Number that is authorized to originate a route announcement for the prefix corresponding to the SRO record's reverse DNS name. If a prefix can be announced from multiple AS numbers, then multiple SRO records should be defined at the domain name corresponding to that prefix.

For example, to authorize AS 12145 as the origin AS for 129.82.0.0/16, one would create an SRO record with Origin ASN 12145 and store the SRO record at the name m.82.129.in-addr.arpa.

The presence of an SRO record authorizes a prefix to be announced and specifies the origin ASN. If there is no SRO record present at the domain name, then one of two cases occur. Either a sub-prefix hijack is occurring, or BGP authentication is not being used in a particular zone. To distinguish between the two cases, the Route Lock (RLOCK) resource record is used to "opt-in" the zone to BGP origin authentication. The RLOCK is placed at the apex of a reverse-DNS zone to indicate that the zone is being used to publish routing information. If this record is present, all route announcements for the prefixes covered by this zone must be authorized by an SRO record.


Example:
publish origins for one /16 and four /18's.
in-addr
129
82
Zone file: (uses CIDR reverse-DNS naming convention)

```
$ORIGIN 82.129.in-addr.arpa
$TTL 3600

@      IN  RLOCK  ; secure entire zone
m      IN  SRO 12145  ;129.82.0.0/16
0.0.m IN  SRO 12145  ;129.82.0.0/18
1.0.m IN  SRO 12145  ;129.82.64.0/18
0.1.m IN  SRO 12145  ;129.82.128.0/18
1.1.m IN  SRO 12145  ;129.82.192.0/18

; can now directly add /24 SROs
; or can let the lower octet do it

; existing delegations

0     IN   NS     rush.colostate.edu
1     IN   NS     rush.colostate.edu
;....
255   IN   NS     rush.colostate.edu
```

RLOCK = Route LOCK (currently using TYPE65400)
SRO = Secure Route Origin (currently using TYPE65401)
Automated provisioning tools have been written

Figure 3.2. *Zone file showing RLOCK and SRO records*

The effective span of control for an RLOCK is dependent on the structure of the Reverse DNS zone. A Reverse DNS zone that has no delegations will have a span of control that covers all prefixes at or below the CIDR prefix specified by the domain name at the zone apex. Any zone delegation starts a new zone authority. Those prefixes in the delegated zone will not be covered by the parent zone's RLOCK.

As an example, consider the zone at 129.82.0.0/16 and assume that it has only one delegation at 129.82.138.0/24. The 129.82.0.0/16 RLOCK covers all prefixes within the /16 to /32 range with the exception of prefixes within the 129.82.138.0/24 through /32 range because of the delegation. The 129.82.138.0/24 administrator could independently choose to provide BGP authentication by adding an RLOCK to the zone apex or could choose to simply not participate.

[13] provides a more comprehensive explanation of the resource records. A sample zone file snippet illustrating both the naming convention and the resource records is illustrated in figure 3.2.

As a final note, Chapter 7 discusses recent modifications to the SRO and RLOCK records that add fields for *flags, activation time*, and *prefix-limit*. These fields were added to handle operational issues that could occur during initial publishing and during delegation of an IP prefix to another owner. The full discussion is in Section 7.2.

3.4. *ROVER Validation Algorithm*

Once an organization has published ROVER data in the reverse DNS, anyone can use the data to authenticate BGP routing announcements. By design, ROVER does not specify how the data will be used. Our work with different operational networks has shown that there are many different models. One example is an application to perform near-real-time route origin monitoring that alerts operators of hijacks. A more aggressive approach directly interacts with a router to detect and automatically block the perceived hijack. Another application could perform a nightly analysis that generates router prefix filters. Other applications may not interact with BGP routes at all; the approaches cross-check data in the Internet Routing Registries (IRR) against the data in the reverse DNS. This list is not intended to be comprehensive, but instead aims to illustrate the potential uses of the published data.

To illustrate how ROVER data can be used for verification, we present an example application that analyzes BGP announcements in real-time as announcements arrive from a BGP data aggregator. The application validates each announcement with a query to the reverse DNS as outlined in *Algorithm 2*. This algorithm classifies route route announcements into one of the following three categories:

(1) *VALID*: a DNSSEC-validated SRO RRSET was received and one of the route origins in the RRSET matches the origin contained in the BGP route announcement.

(2) *INVALID*: a route hijack was detected because either 1) the DNSSEC-validated SRO responses received did NOT match the origin of the route announcement (e.g. an origin hijack) or 2) there was no SRO record at the domain name corresponding to this prefix, but the authoritative zone did contain an RLOCK statement (e.g. a sub-prefix hijack).

(3) *NOTFOUND*: there was no SRO record for this prefix and no RLOCK record to protect the zone, or the data did not properly validate with DNSSEC. In this case, the algorithm cannot authoritatively state that the prefix is valid or invalid, so it is simply marked as not found.

The actual DNS query process is simple. Upon receiving a route announcement, ROVER converts the prefix into a DNS name (as discussed above) and issues a query for the SRO records at that name. If DNSSEC authenticated SRO records are returned, the route is classified as either VALID if the origin ASN is authorized or INVALID if the origin does not match.

If no records are returned (NXDOMAIN or NOERROR with number of answers=0), the authority section of the DNS response identifies the covering zone. ROVER then performs a query to that domain name (the zone apex) for an RLOCK record. If an RLOCK is returned, the zone owner has deployed ROVER. The DNSSEC authenticated denial of existence for the SRO proves the zone owner did not authorize route announcements for the prefix. This is a sub-prefix hijack and the prefix is marked as INVALID. If no RLOCK exists, the zone owner has not deployed ROVER and the prefix is marked as NOTFOUND.

This verification algorithm is "fail-safe". If a query for a DNS record fails, or if DNSSEC fails to validate the record, the algorithm behave as if no DNS records were present in the first place.

Algorithm 2 The ROVER prefix validation algorithm. This function compares the announced BGP origin with authoritative data retrieved from the reverse DNS and returns VALID, BOGUS or UNKNOWN. Origin and sub-prefix hijacks are both detected.

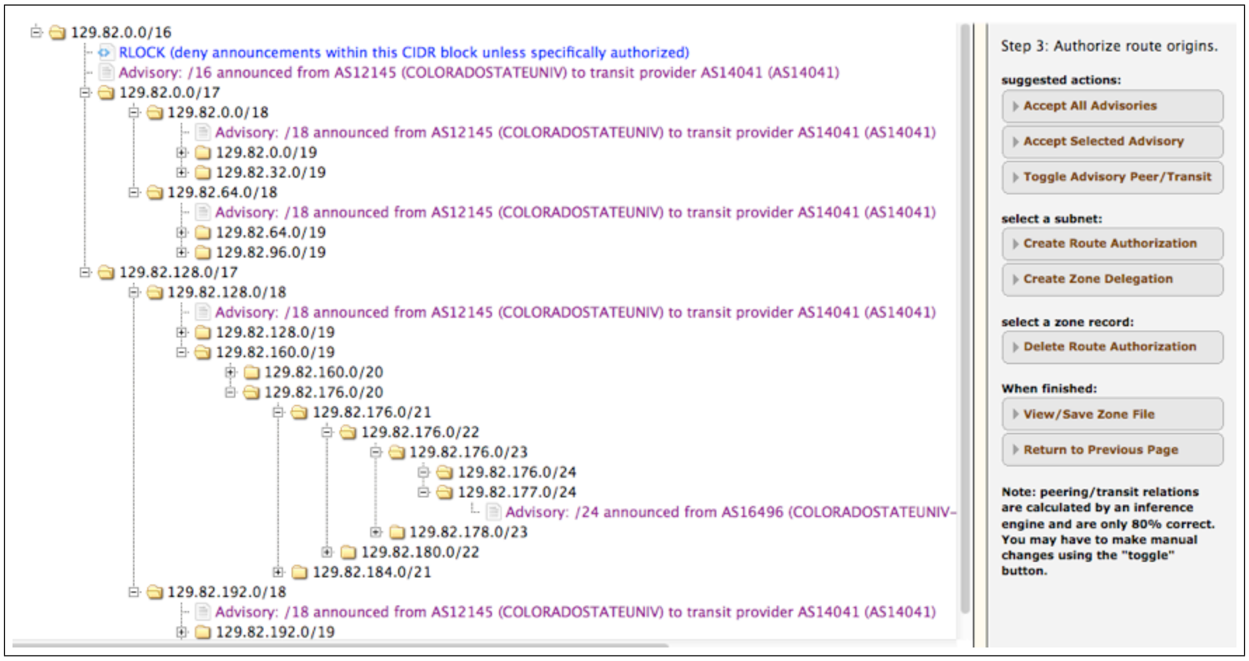
```
function VALIDATE( $prefix, origin$)
    $DNSname \leftarrow convertToReverseDNS(prefix)$                    // convert to DNS name
    $response, rc, flags \leftarrow doDNSquery(DNSname, SRO)$            // query the DNS
    if $rc = TIMEOUT$ then return UNKNOWN                             // no response
    if not ADbit in flags then return UNKNOWN                  // response didn't have DNSSEC
    if $rc = NOERROR$ and $length(response) \neq 0$ then            // got non-empty response?
        for all RRset in response do
            if $RRset_{type} = SRO$ then                            // loop through SROs received
                for all RRData in RRset do
                    if $origin = RRdata_{origin}$ then return VALID           // Origin is Valid!
                return BOGUS                         // Origin Hijack! None of the SROs matched
    if $rc = NXDOMAIN$ or $rc = NOERROR$ then                  // No SROs. Check for RLOCK.
        $DNSname = response_{authoritySection}$                              // find zone apex
        if $DNSname =$ in-addr.arpa, etc. then return UNKNOWN          // don't search too far
        $response, rc, flags \leftarrow doDNSquery(DNSname, RLOCK)$          // query the DNS
        if $rc = TIMEOUT$ then return UNKNOWN                             // no response
        if not ADbit in flags then return UNKNOWN              // response didn't have DNSSEC
        if $rc = NOERROR$ and $length(response) \neq 0$ then return BOGUS
                                          // RLOCK exists. Sub-prefix hijack detected.
        return UNKNOWN                             // no SRO, no RLOCK found. Zone opt-out.
    else return UNKNOWN                   // SERVFAIL, DNSSEC didn't validate, FORMERR, etc.
```

This results in marking a BGP announcement as *NOTFOUND*. A successful DNSSEC-downgrade attack would result in classifying records as NOTFOUND. However the redundancy in DNS would allow checking of multiple slave DNS servers should DNSSEC fail to validate.

3.5. *Example Application: The ROVER Testbed*

A web-accessible testbed, `rover.secure64.com` [9], has been built as a both a proof-of-concept and as a platform for experiments with origin validation.

After creating an account on the testbed web site, a user can experiment with several different functional areas. The user can define and publish a zone file containing SRO and RLOCK records, perform a ROVER route verification, or view a live IP hijack monitor which examines real-time data from BGPmon. In this section we will only discuss the publisher and hijack monitor.



```
129.82.0.0/16
  RLOCK (deny announcements within this CIDR block unless specifically authorized)
  Advisory: /16 announced from AS12145 (COLORADOSTATEUNIV) to transit provider AS14041 (AS14041)
  129.82.0.0/17
    129.82.0.0/18
      Advisory: /18 announced from AS12145 (COLORADOSTATEUNIV) to transit provider AS14041 (AS14041)
      129.82.0.0/19
      129.82.32.0/19
    129.82.64.0/18
      Advisory: /18 announced from AS12145 (COLORADOSTATEUNIV) to transit provider AS14041 (AS14041)
      129.82.64.0/19
      129.82.96.0/19
  129.82.128.0/17
    129.82.128.0/18
      Advisory: /18 announced from AS12145 (COLORADOSTATEUNIV) to transit provider AS14041 (AS14041)
      129.82.128.0/19
      129.82.160.0/19
        129.82.160.0/20
        129.82.176.0/20
          129.82.176.0/21
            129.82.176.0/22
              129.82.176.0/23
                129.82.176.0/24
                129.82.177.0/24
                  Advisory: /24 announced from AS16496 (COLORADOSTATEUNIV-
              129.82.178.0/23
            129.82.180.0/22
          129.82.184.0/21
    129.82.192.0/18
      Advisory: /18 announced from AS12145 (COLORADOSTATEUNIV) to transit provider AS14041 (AS14041)
      129.82.192.0/19
```

Step 3: Authorize route origins.

suggested actions:

select a subnet:

select a zone record:

When finished:

Note: peering/transit relations are calculated by an inference engine and are only 80% correct. You may have to make manual changes using the "toggle" button.

Figure 3.3. *Secure64 ROVER Testbed showing the provisioning system. Historical routes are displayed for a prefix and the user may choose to accept, reject or modify them. A zone file containing the appropriate RLOCK and SRO records is then automatically generated.*

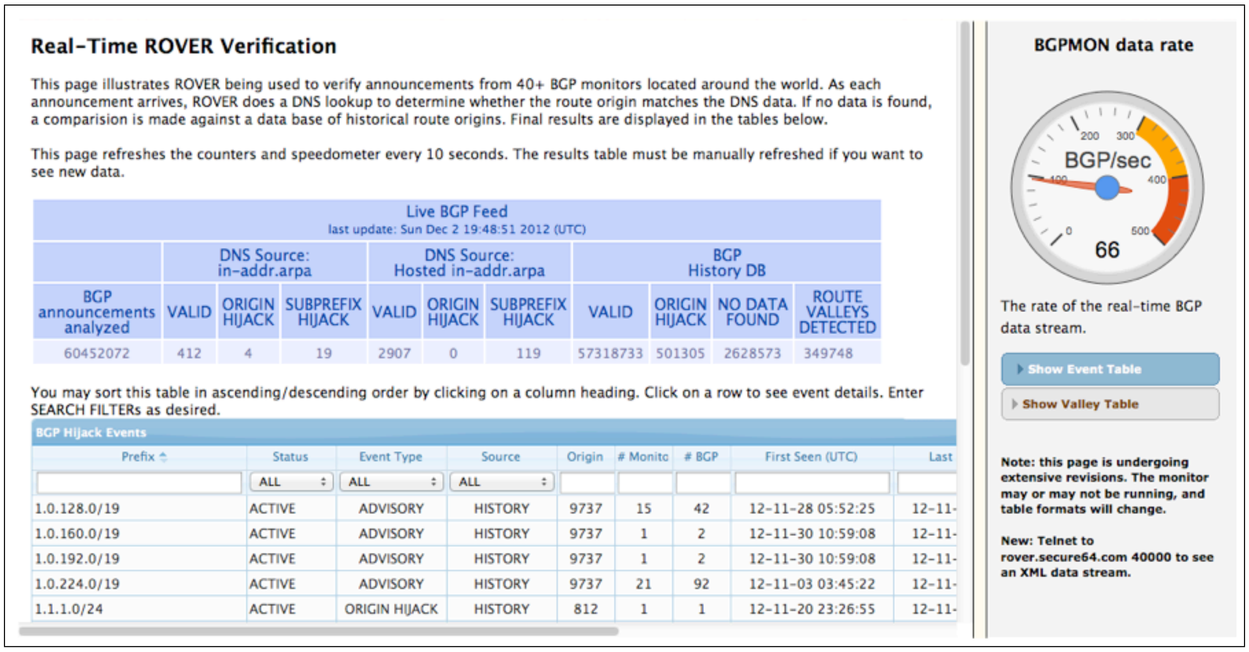
3.5.1. *Zone Provisioning.* ROVER data is normally published in reverse DNS zones. Several organizations have already done so. Although simply adding names and records will not break existing DNS deployments, most organizations have policies and provisioning systems for managing the DNS. Obtaining authorization to add records to the DNS may be non-trivial and updating provisioning systems can be time consuming. We do not expect that typical operators will invest time in understanding how to convert an IP prefix into the DNS name, nor how to generate the SRO and RLOCK statements. Scripts, tools, services, and documents are all needed. As a lower barrier to entry, the ROVER testbed provides a provisioning system and a shadow zone in which one can experiment.

The provisioning tool automatically detects existing BGP announcements used by an ISP and converts them into a ROVER zone file. The program retrieves WHOIS data, looks at a history data base to show previously seen BGP announcements related to that net block, and uses the CAIDA AS-inference table to present the transit providers associated with each origin. The operator can accept the suggested route announcements as is, delete some, add others, and then generate zone file contents by a simple button push. A sample screen shot of the provisioning system is illustrated in figure 3.3

We collaborated with both large and small ISPs in constructing the provisioning tool. Operators were able to provision zones on their own. The provisioning tool is useful for small ISPs and has been used by large ISPs to provision a portion of their addresses. It is impractical for large organizations with hundreds or thousands of net blocks. We foresee the need for additional tools that can generate ROVER zones from reading router configuration files or obtaining data from other information sources owned by the ISP. Several operators who were completely unknown to us were also able to provision zones with no assistance or information other than what was present on the web site. These zones just "showed up" as various ISPs in Europe, America and Australia heard about ROVER and signed up on the web site to try their own experiments.

3.5.2. *IP Hijack Monitor.* Once data has been provisioned, ROVER uses a real-time BGP feed provided by RouteViews/BGPmon [48]. As the RouteViews peer announces a route change, the change is propagated to ROVER in real-time. ROVER receives data from roughly 40 peers; the number varies slightly as the real-time feed is being expanded by the RouteViews and BGPmon teams. ROVER classifies each update as VALID, INVALID, or NOTFOUND as described in the previous section.

Figure 3.4 shows a screen capture from the ROVER Verification interface. The speedometer on the right shows the rate of incoming BGP data messages from the RouteViews peers. Every prefix is checked against three distinct data sources. First, ROVER queries the existing reverse DNS.


Real-Time ROVER Verification

This page illustrates ROVER being used to verify announcements from 40+ BGP monitors located around the world. As each announcement arrives, ROVER does a DNS lookup to determine whether the route origin matches the DNS data. If no data is found, a comparision is made against a data base of historical route origins. Final results are displayed in the tables below.

This page refreshes the counters and speedometer every 10 seconds. The results table must be manually refreshed if you want to see new data.

Live BGP Feed last update: Sun Dec 2 19:48:51 2012 (UTC)										
	DNS Source: in-addr.arpa			DNS Source: Hosted in-addr.arpa			BGP History DB			
BGP announcements analyzed	VALID	ORIGIN HIJACK	SUBPREFIX HIJACK	VALID	ORIGIN HIJACK	SUBPREFIX HIJACK	VALID	ORIGIN HIJACK	NO DATA FOUND	ROUTE VALLEYS DETECTED
60452072	412	4	19	2907	0	119	57318733	501305	2628573	349748

You may sort this table in ascending/descending order by clicking on a column heading. Click on a row to see event details. Enter SEARCH FILTERs as desired.

BGP Hijack Events

Prefix	Status	Event Type	Source	Origin	# Monito	# BGP	First Seen (UTC)	Last
	ALL	ALL	ALL					
1.0.128.0/19	ACTIVE	ADVISORY	HISTORY	9737	15	42	12-11-28 05:52:25	12-11-
1.0.160.0/19	ACTIVE	ADVISORY	HISTORY	9737	1	2	12-11-30 10:59:08	12-11-
1.0.192.0/19	ACTIVE	ADVISORY	HISTORY	9737	1	2	12-11-30 10:59:08	12-11-
1.0.224.0/19	ACTIVE	ADVISORY	HISTORY	9737	21	92	12-11-03 03:45:22	12-11-
1.1.1.0/24	ACTIVE	ORIGIN HIJACK	HISTORY	812	1	1	12-11-20 23:26:55	12-11-

BGPMON data rate

The rate of the real-time BGP data stream.

Note: this page is undergoing extensive revisions. The monitor may or may not be running, and table formats will change.

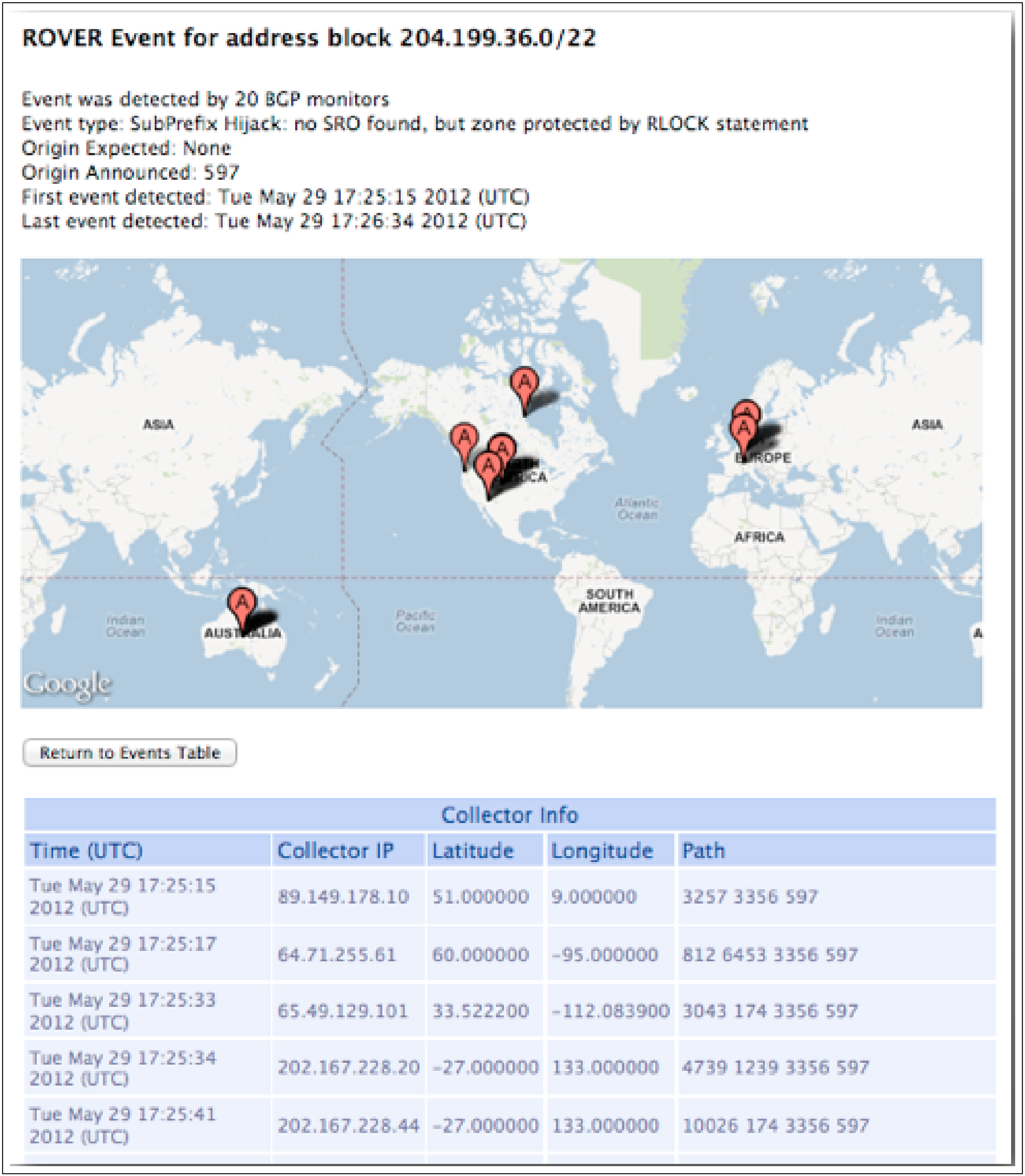
New: Telnet to rover.secure64.com 40000 to see an XML data stream.

Figure 3.4. *Secure64 Testbed showing ROVER & History Attack Detection*

As the Figure shows, only a small number of prefixes currently have data entered in the existing reverse DNS. Second, Rover queries the shadow reverse DNS hosted at at the ROVER site. Finally, ROVER keeps a historical log of all BGP updates and compares each update against the historical record. The table at the bottom of Figure 3.4 show each potential hijack that was detected.

By clicking on the event, a user can determine which BGP routers have observed the event and exam other details such as the actual BGP updates associated with the event. Figure 3.5 shows the detailed view of one event. The map indicates the geographic location of BGP routers who observed the potential hijack. One can click on the BGP peer router or the row to show the actual BGP updates associated with the event.

To provide ground truth, we worked with a tier-1 ISP to both publish data and announce an intentionally invalid route. The ISP published their DNSSEC-signed reverse-DNS. The ISP announced a sub-prefix hijack at a time chosen at random. This single BGP announcement was one of a multitude of announcements that the ROVER testbed analyzed on a continuous basis. The



ROVER Event for address block 204.199.36.0/22

Event was detected by 20 BGP monitors
Event type: SubPrefix Hijack: no SRO found, but zone protected by RLOCK statement
Origin Expected: None
Origin Announced: 597
First event detected: Tue May 29 17:25:15 2012 (UTC)
Last event detected: Tue May 29 17:26:34 2012 (UTC)

Collector Info				
Time (UTC)	Collector IP	Latitude	Longitude	Path
Tue May 29 17:25:15 2012 (UTC)	89.149.178.10	51.000000	9.000000	3257 3356 597
Tue May 29 17:25:17 2012 (UTC)	64.71.255.61	60.000000	-95.000000	812 6453 3356 597
Tue May 29 17:25:33 2012 (UTC)	65.49.129.101	33.522200	-112.083900	3043 174 3356 597
Tue May 29 17:25:34 2012 (UTC)	202.167.228.20	-27.000000	133.000000	4739 1239 3356 597
Tue May 29 17:25:41 2012 (UTC)	202.167.228.44	-27.000000	133.000000	10026 174 3356 597

Figure 3.5. *Rover Event showing geographical detectors*

hijack was successfully detected in less than a minute and observed to propagate across the world in a span of five minutes.

The results also shows the advantage of using multiple vantage points. BGPMon collects route announcements from multiple world-wide BGP peers. Multiple vantage points can expose a localized or stealthy IP hijack. The testbed hijack detector displays a world-wide map indicating which collectors have encountered the hijack, and which are operating normally. This is very useful to an E-commerce or other organization which may think that everything is operating perfectly, but is blissfully unaware that no one in Europe or other parts of the world can reach their IP block.

CHAPTER 4

GLOBAL INTERNET CHARACTERISTICS AND MEASUREMENTS

With ROVER defined, we now proceed to the research questions outlined in the introduction to this dissertation: how well does ROVER work? How many organizations have to deploy it in order for it to be effective at detecting or preventing hijacks? What are its issues and limits?

We perform this analysis in the next four chapters. To set the stage, this chapter describes baseline metrics of the Internet topology that are relevant to IP hijack propagation. Chapter 5 gives an analysis of IP hijack propagation in the absence of any defenses. Chapter 6 examines the effectiveness of ROVER within an incremental deployment model. Finally, Chapter 7 probes the vulnerabilities and dependencies of ROVER. We examine its resilience in the face of threat or component failure, and measure the load it poses on the global DNS system.

In this chapter we discuss measurements and characteristics of the global Internet. Section 4.1 describes the simulation program developed to perform measurements and experiments. Section 4.2 discusses the accuracy of the model with respect to the real-world Internet. Finally, Section 4.3 explores baseline metrics for the simulated topology We introduce two new metrics, *depth* and *reach*, that will be seen to correlate well with the success or failure of an IP hijack attempt.

4.1. *A Simulation Model of the Global Internet*

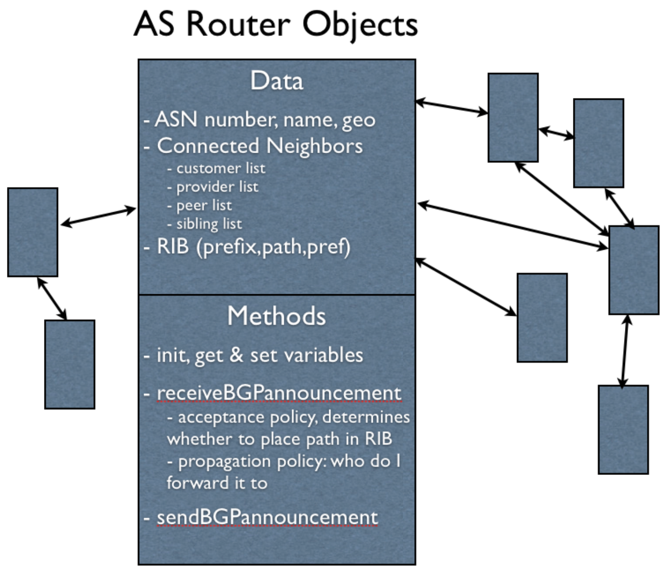
The Global Internet simulator is a small object-oriented Python program that performs BGP announcements and attacks within a model reflecting the structure of the real Internet. This program was developed to meet the specific needs of this thesis. Other available simulation programs were examined for use but were found to be unsuitable. Those simulator programs focused on message passing between two routers and verification of the BGP protocol itself. We needed a program that would quickly propagate BGP announcements and manage the relationships among tens of thousands of routers. The end result was a program that could fully propagate an announcement throughout the simulated topology within 2 seconds on an x86 based computer. Even this was slow when a test run had to loop through thousands of announcements and attacks. Several of the graphs in Chapter 6, for example, took 24 hours of computation for just one line. 7 days of computation were required to plot one graph.

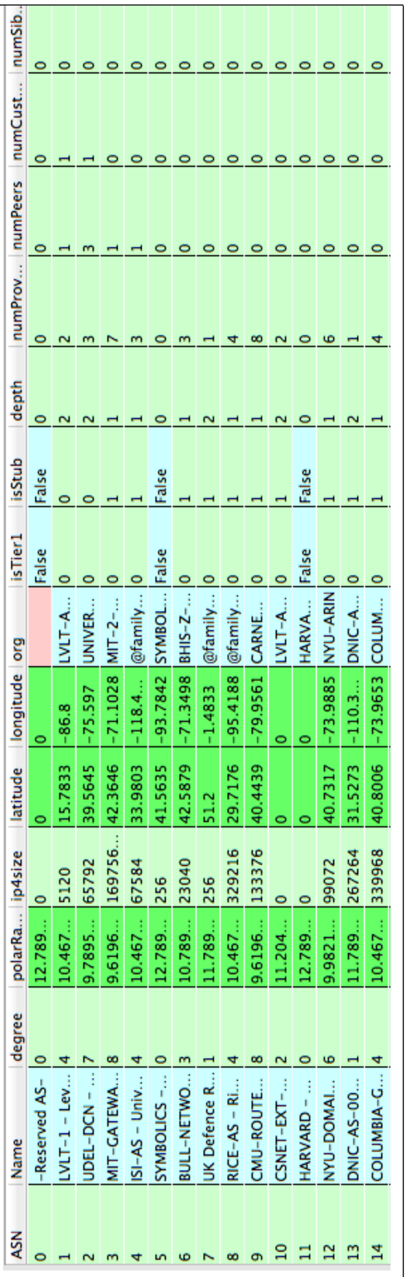
On startup the simulator constructs a topology of 42,697 interconnected router objects as it reads a list of 139,156 relationships. These relationships define whether a neighboring AS is a provider, peer, customer or sibling. The relationship table was obtained from CAIDA file `20121101.as-rel.txt` [6].

The simulator program is complemented with a SQLITE3 database containing useful information about each individual autonomous system. This supplemental database (see Figure 4.1) lists the AS number, name, degree of connectedness, number of IP addresses owned by the AS, geolocation information, and organizational owner. The ASINFO table was constructed in multiple passes by combining various data sources including routing tables from Oregon ROUTEVIEWS [38], the CAIDA relationship table, CAIDA organization table (`as_org2info.20120411.txt`, a work-in-progress by CAIDA's Bradley Huffaker), and geolocation data downloaded from *maxmind.com* [30].

Each router object (Figure 4.1) contains its own local state information:

- inter-connection tables for peers, customers, transit providers and siblings
- AS name and geolocation data
- a RIB table to store prefix announcements, AS paths, and LOCAL_PREF used





ASN	Name	degree	polarRa...	ip4size	latitude	longitude	org	isTier1	isStub	depth	numProv...	numPeers	numCust...	numSib...
0	-Reserved AS-	0	12.789...	0	0	0		False	False	0	0	0	0	0
1	LVLT-1 - Lev...	4	10.467...	5120	15.7833	-86.8	LVLT-A...	0	0	2	2	1	1	0
2	UDEL-DCN - ...	7	9.7895...	65792	39.5645	-75.597	UNIVER...	0	0	2	3	3	1	0
3	MIT-GATEWA...	8	9.6196...	169756...	42.3646	-71.1028	MIT-2-...	0	1	1	7	1	0	0
4	ISI-AS - Univ...	4	10.467...	67584	33.9803	-118.4...	@family...	0	1	1	3	1	0	0
5	SYMBOLICS -...	0	12.789...	256	41.5635	-93.7842	SYMBOL...	False	False	0	0	0	0	0
6	BULL-NETWO...	3	10.789...	23040	42.5879	-71.3498	BHIS-Z-...	0	1	1	3	0	0	0
7	UK Defence R...	1	11.789...	256	51.2	-1.4833	@family...	0	1	2	1	0	0	0
8	RICE-AS - Ri...	4	10.467...	329216	29.7176	-95.4188	@family...	0	1	1	4	0	0	0
9	CMU-ROUTE...	8	9.6196...	133376	40.4439	-79.9561	CARNE...	0	1	1	8	0	0	0
10	CSNET-EXT-...	2	11.204...	0	0	0	LVLT-A...	0	1	2	2	0	0	0
11	HARVARD - ...	0	12.789...	0	0	0	HARVA...	False	False	0	0	0	0	0
12	NYU-DOMAI...	6	9.9821...	99072	40.7317	-73.9885	NYU-ARIN	0	1	1	6	0	0	0
13	DNIC-AS-00...	1	11.789...	267264	31.5273	-110.3...	DNIC-A...	0	1	2	1	0	0	0
14	COLUMBIA-G...	4	10.467...	339968	40.8006	-73.9653	COLUM...	0	1	1	4	0	0	0

Figure 4.1. *Simulator Object Model and Data Base*

Each router object has a set of methods to perform housekeeping tasks, but more importantly, to make and to receive prefix announcements with its neighbors. Each router object follows a strict policy for these methods to ensure "best path acceptance" and "valley-free routing" are enabled.

PREFIX ACCEPTANCE POLICY: A router will accept an announcement and place it into its RIB based on LOCAL_PREF and shortest path. LOCAL_PREF is set such that customers are preferred over peers, and peers are preferred over transit providers. If a router already has an announcement in its RIB and a new announcement arrives with equal LOCAL_PREF, then the new announcement is accepted only if has a shorter path length than the previously accepted announcement.

PROPAGATION POLICY: Egress filters are used that enforce valley-free routing policy. Most ISPs, but not all, follow valley-free routing rules:

- customer to provider: export all prefixes except those of its providers and peers
- provider to customer: export all routes
- peer to peer: export customer and sibling only
- sibling to sibling: uses a community string to create the equivalent of one AS out of multiple sibling ASes, then propagates messages according to the rules above.

The simulation is capable of propagating announcements and attacks among the inter-connected routers and writes relevant statistics into the database. Results can then be visualized with graphics and tables. Some of the graphic programs developed were inspired by the CAIDA website of internet topology visualizations [7]. These visualizations are useful for gaining insights on attack propagation, especially when comparing before and after scenarios to see the effect or ROVER and how attacks might still propagate despite defensive measures.

The polar graph in Figure 4.2 shows a visualization of the 42,696 ASes contained in the simulated topology.

- Longitude of any AS is plotted along the circular perimeter. One can clearly see AS densities in Europe, Asia and the Americas, with corresponding sparseness in ocean areas.
- Radius from the center determines the degree of connectivity of an AS based on a logarithmic scale. The closer to the center the higher the number of interconnections. Tier1

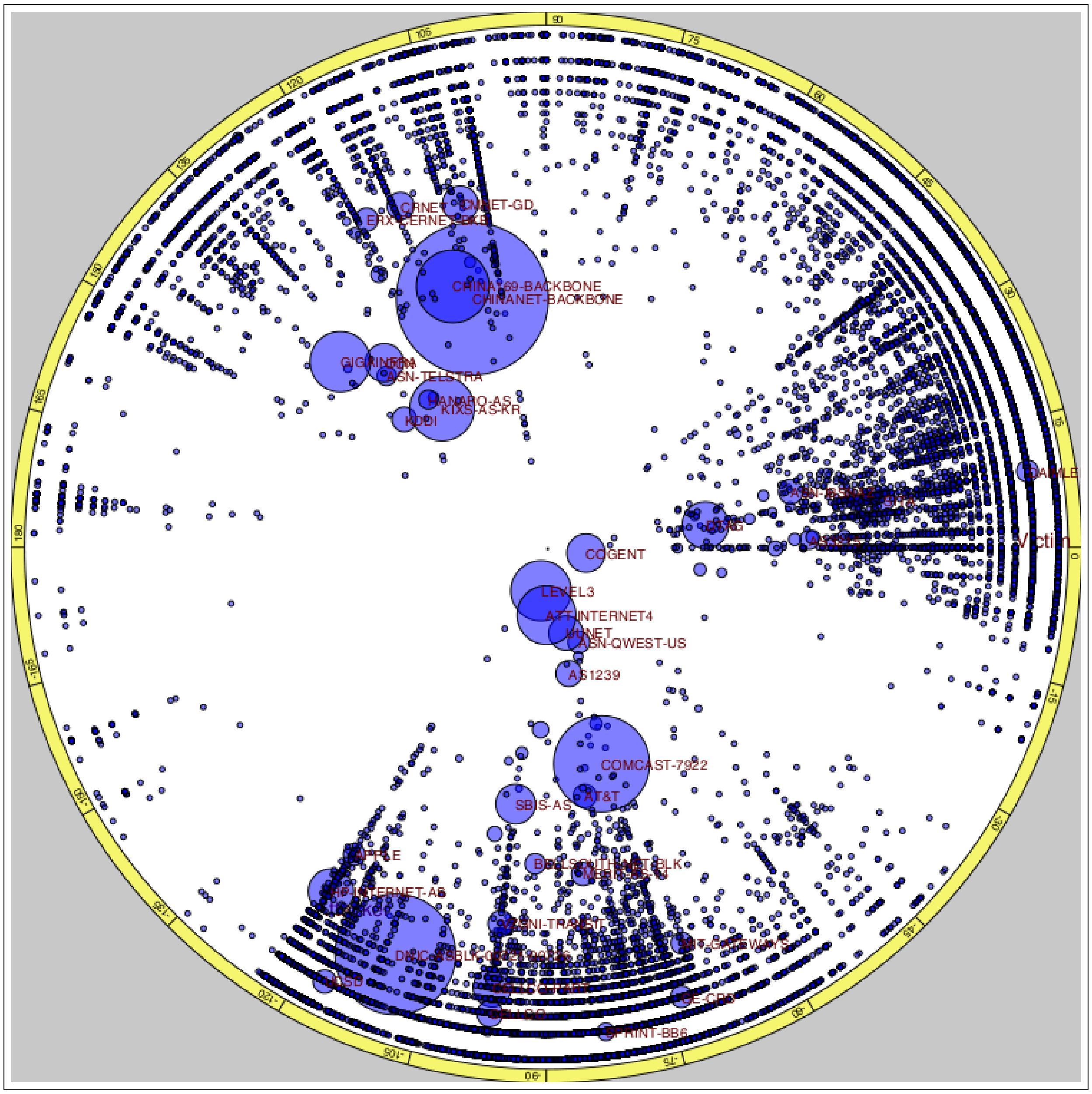


Figure 4.2. *Polar graph of Network Topology*

ASes appear in the middle. Cogent, for example, has 3738 interconnected ASes. The further out towards the perimeter, the smaller the degree of AS interconnections. The outer perimeter ASes have only 1 or 2 interconnections.

- Finally, the diameter of each circle gives an indication of the total size of the address space assigned to each AS. Large address space is owned by ASes such as China Telecomm,

```
enter a command:  announce 25883

AS 25883 announcing 25883
generation 0:  number of messages=3
generation 1:  number of messages=5287
generation 2:  number of messages=44286
generation 3:  number of messages=30394
generation 4:  number of messages=4410
generation 5:  number of messages=349
generation 6:  number of messages=2
announcement complete; propagated to 42320 ASes
```

Figure 4.3. *Simulator output from* `announce 25883` *command*

the United States Department of Defense, Cogent, Level3, Sprint, HP, Apple, etc.

The simulator also has a command language interpreter (CLI). After startup the operator can issues commands such as "`announce 12145`" to initiate a BGP announcement that will propagate throughout the topology. The command "`attack 6582 12145`" will cause AS6582 to perform an IP hijack against AS12145. Other commands exist to flush the RIB tables, find longest or nonexistent paths, display summary statistics for the topology, and to print the state of any particular RIB. For example, the output from "`announce 25883`" is shown in Figure 4.3.

This example illustrates an important point: BGP messages are propagated to neighboring routers one generation at a time, just like the real internet. A router will examine all the messages it has received during one generation and then propagate the best choice to its neighbors at the next generation of announcements. Propagation stops when there are no messages left to send.

The simulator has useful commands to display statistics and the current state of any particular AS in the topology. Figure 4.4 shows the `print` command for the Colorado State University AS (12145) after it has received the announcement from 25883. Note the use of arrows in the *RIB relationship* line. These indicate a customer-to-provider or provider-to-customer relationship. The

```
enter a command:  print 12145

ASN 12145:  Colorado State University
Address Space:  66560 IPv4 addresses
Degree:  5
latitude:  40.088 longitude:  -105.374

providers:  1299, 14041

customers:  25825, 31991, 54060

peers:

siblings:

RIB:
 25883:  pref=100 Path:[14041, 3356, 25883]
 Relationships:  12145 --> 14041 --> 3356 <-- 25883
 12145:  COLORADOSTATEUNIV - Colorado State University
 14041:  University Corp for Atmospheric Research
 3356:  LEVEL3 Level 3 Communications
 25883:  CITIGROUP - Citigroup
```

Figure 4.4. *Simulator output from* `print 12145` *command*

tip of the arrow points to the provider. This styling enables one to see that there there is always a customer to provider to customer chain, and that no valleys exist. Other symbols are used for tier-1 to tier-1 relationships, peer-to-peer relationships, and sibling-to-sibling relationships.

We now give an example showing how simulation results can be used to generate a series of graphics or animated movies that visually display an attack.

The polar graphs in Figure 4.5 illustrate a sequence of BGP announcements in which AS6582 (Front Range Internet based in Fort Collins, CO) is hijacking the IP Prefix of AS25883 (Citigroup, chosen simply because it is an attractive hijack target – a bank).

The attack begins with AS6852 sending the prefix to its 15 BGP neighbors. Some of the neighbors reject the announcement (lines shown in green) because of policy. Their existing paths

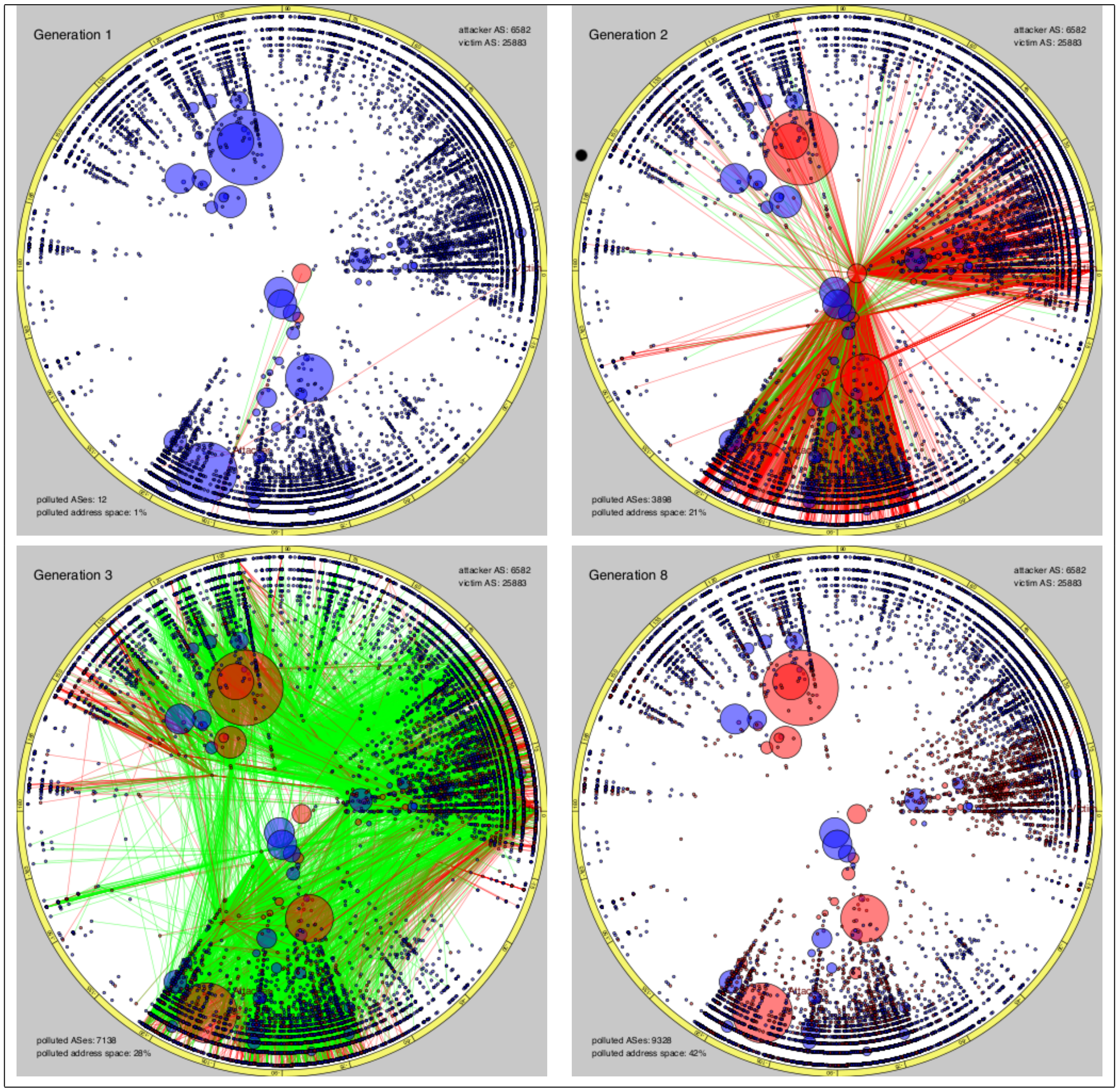


Figure 4.5. *AS6582 (FRII) hijacking AS25883 (CitiGroup)*

may be shorter or may have a higher LOCAL_PREF. Twelve of the neighbors accept the "better path" however (line shown in red). In particular, a tier-1 transit provider, Cogent, has accepted the announcement.

The second generation begins after a nominal 30 second router timeout. A veritable hailstorm of announcements is sent out from several radius points. Most of the lines are red, causing further

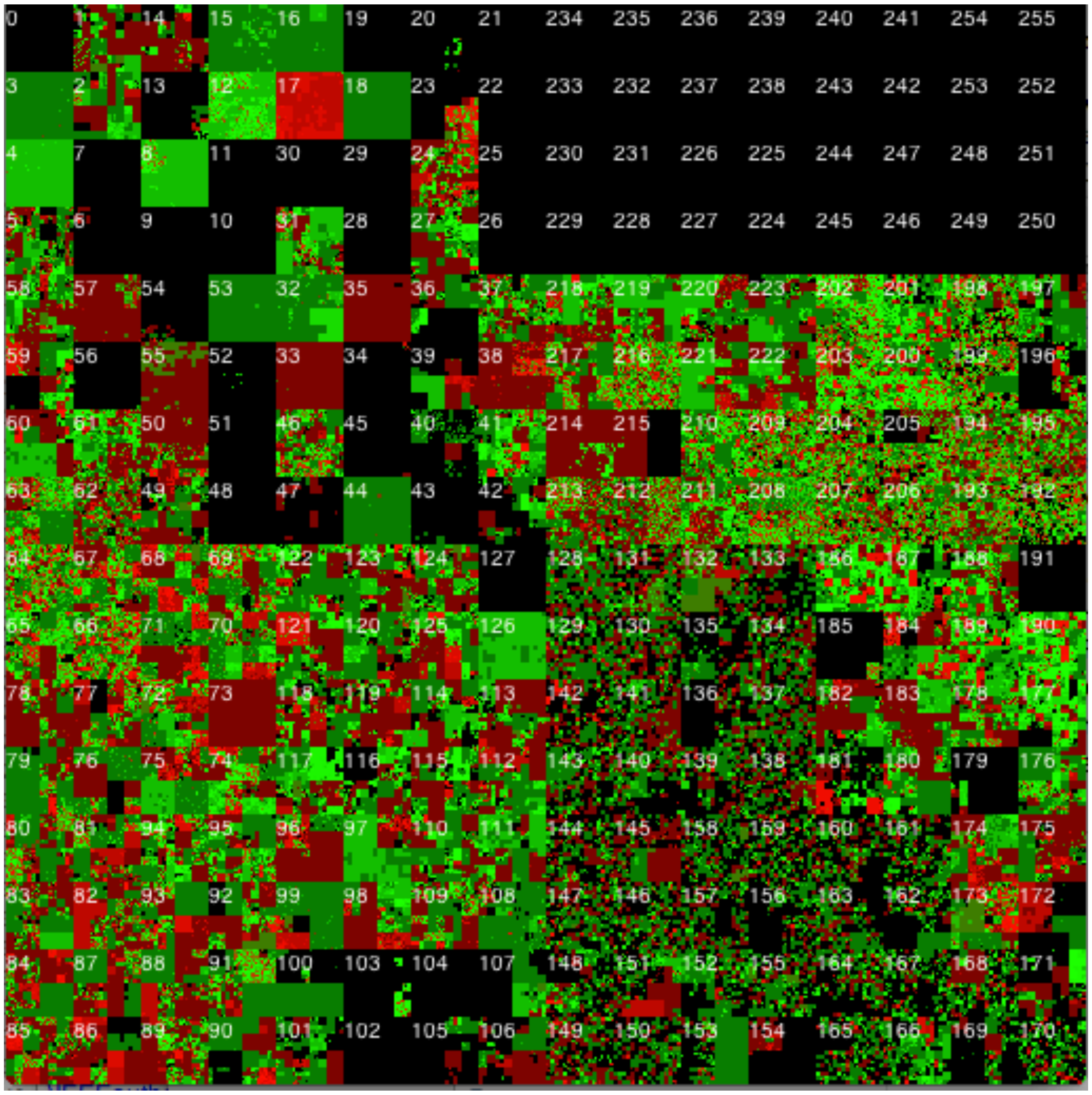


Figure 4.6. *Hilbert Graph; red areas cannot reach AS25883*

AS pollution. 3898 routers are polluted at the end of this step; 21% of the internet address space can no longer reach Citigroup.

The third generation continues, but in this case most of the announcements are rejected. The paths are starting to get long. Nevertheless we now have more than 7000 ASes polluted and 28% of the address space cannot reach Citigroup.

This continues until BGP convergence is reached in the 8th generation (about 4 minutes). In the end, 9328 ASes are polluted and 42% of the internet address space cannot reach Citigroup. This graphic visualization shows both large and small ASes have been impacted, distributed worldwide.

Visualization can also be used to illustrate the extent of an attack. The graph shown in Figure 4.6 shows a map of the entire IPv4 address space organized into blocks following a Hilbert path through 2D space. Each of the 256 /8 address blocks are clearly defined and smaller spaces exist as fractals within each of these blocks. Green blocks are untouched by the IP hijack and can still reach Citigroup. Red blocks are polluted and will be routed to the attacker instead of the proper destination. These red blocks can be seen to be distributed across both large and small IP address blocks.

4.2. *Simulation Accuracy*

A natural concern with any research based on simulation is whether the experimental results obtained are an accurate reflection of what would happen in the real Global Internet.

Other papers, notably [25] and [16] reference experimental results obtained from simulators that are similar to ours. In [16], the authors caution that *"Because we work within a model of routing policies, we caution against interpreting our results as hard numbers that measure the impact of an attack launched by a specic manipulator in practice. However, the trends uncovered by our quantitative analysis do allow us to arrive at a number of useful insights."*.

Complete accuracy is not a practical goal, nor is it necessarily needed. The `BGPSIM` system developed by NLNetLabs [34] was designed to balance tradeoffs between scalability, CPU efficiency, extensibility and accuracy. The authors state *" For global operation and stability of BGP, it is questionable whether an "exact" model of the AS topology is needed. An interesting direction of research might be to identify classes of similar topology for which results are generalizable."* In

Table 4.1. *Comparison of Five Simulation Runs to RIB Entries from Oregon RouteViews*

	deterministic with tier-1 valley-free	deterministic with tier-1 shortest path	random-10 with tier-1 valley-free	random-2 with tier-1 valley-free	random-2 with tier-1 shortest-path
# routes	1,186,568	1,186,568	1,186,568	1,186,568	1,186,568
match	425,501	473,275	246,624	282,806	372,392
all but 1	248,752	278,330	168,383	196,024	250,100
total ok	674,253	751,605	415,007	478,830	622,492
percent ok	56.8%	**63.3%**	35.0%	40.4%	52.5%

that spirit, my simulations could have been run on a completely synthetic topology and yet would still generate significant learnings. Furthermore, the real internet's AS topology evolves daily so that results start to become less accurate over time.

Nevertheless, one would want the simulation to at have a as strong a correspondence as possible to the actual Internet so that real-world comparisons could be made. To that end, a series of modifications were made to the simulation engine and measured for accuracy. Most of these modifications involved changes to the internet topology (AS relation table), BGP policy enforcement, and use of Monte-Carlo randomness techniques.

The various modified topologies were compared path-by-path with over 1 million routes taken from the Oregon RouteViews (file `rib.ASCII.20130111`). Simulated paths that either matched the RIB data identically or that had a matching path length with only one AS substitution were considered to be accurate. In the latter case the paths were considered accurate because they were topologically equivalent, for example, substituting provider 1 with provider 2.

The end result, illustrated in table 4.1 showed accuracy ranging from a minimum of 35% to a best-case of 63.3%. This best-case simulator was then used for all attack and defense experiments.

The five variants listed in each column of table 4.1 have the following characteristics and differences:

- **Deterministic Tier-1 valley-free and Deterministic Tier-1 shortest-path**: These two columns have the best accuracy. Paths are selected deterministically, as described by the PREFIX ACCEPTANCE RULES in Section 4.1. The difference between these two models is the treatment of Tier-1 ASes: in one case we strictly follow the valley-free rule at the detriment of path length. In the second case a Tier-1 AS will ignore the valley-free rule if a shorter path is presented to it. This *shortest-path* rule improved accuracy from 56.8% to 63.3%. Here is an example of a Tier-1 using the valley-free rule in which AS2914 selects a path from customer AS5580 because it has a higher LOCAL_PREF than the path from a Tier-1 peer:

```
Path:[5580, 16128, 6]

 Relationships:  2914 <-- 5580 <-- 16128 <-- 6

 2914:  NTT-COMMUNICATIONS

 5580:  ATRATO Atrato IP Networks

 16128:  AGARIK-BULLPI-NETWORK AGARIK

 6:  BULL-NETWORK
```

Compare this to the case following the Tier-1 shortest-path rule:

```
Path:[209, 6]

 Relationships:  2914 =T= 209 <-- 6

 2914:  NTT-COMMUNICATIONS

 209:  ASN-QWEST-US NOVARTIS-DMZ-US

 6:  BULL-NETWORK
```

Thousands of similar paths were shortened using this rule, resulting in overall improved accuracy.

- **Monte-Carlo Experiments** The latter three columns were an attempt to see if adding randomization to path selection would improve accuracy. Surprisingly, the end results were worse, but can be explained logically.

  An AS can have hundreds of connections to other ASes. The simulation algorithm sets LOCAL_PREF to 300 for all customer connections, 200 for all peers, and 100 for all providers. Once a path is placed is in the RIB it will not be replaced until an announcement with a higher LOCAL_PREF or an announcement with an equal LOCAL_PREF but shorter path length is received. This creates the deterministic model described earlier.

  The first set of Monte-Carlo experiments were run in which the LOCAL_PREF assign to an AS was randomized. As table 4.1 indicates, accuracy dropped from 63.3% to as little as 35%. This is because LOCAL_PREF trumps SHORTEST PATH. Take the case of a peer connection. If LOCAL_PREF is randomly spread from 200 to 201, or more widely from 200 to 210, the algorithm will choose the highest LOCAL_PREF even if it already has a peer with a shorter path. This caused paths to lengthen in thousands of cases. The end result was a loss of accuracy.

  To avoid this situation, a different set of Monte-Carlo experiments were then performed. In these experiments the path selection algorithm randomly chose to keep or replace an existing path with an announcement that had identical length and preference. The end result had very similar accuracy, 63.0%, to the deterministic case, 63.3%. The only noticeable difference was that IP hijack simulations had very small variance in the number of polluted ASes. An attack from AS6582 to AS25883, for example, polluted 8139 ASes in the deterministic case. In the Monte-Carlo case, this number varied by only plus-or-minus 11. This was too small an effect to justify the extra computation time and simulation complexity. Consequently, it was decided to continue the research experiments with the deterministic algorithm rather than a randomized one.

Several other items also affect simulation accuracy. CAIDA's AS relationship table is known to be reasonably accurate, but not perfect. Several AS numbers are missing. There are 70 stub ASes that have only a peer connection, but do not have a provider. Sibling relationships are not listed in the table and are instead presented as peers. This caused some connectivity problems. Surprisingly, Stanford University showed limited connectivity because it is isolated by a peer that should be a sibling. Many Department of Defense ASes had similar problems. Therefore I converted 387 peers to siblings to improve accuracy.

Private peers are not contained in the AS relationship table because they are not visible to most routers. Furthermore, some Internet Exchange (IX) route-servers are also not listed in the table. Instead, a direct peer-to-peer relationship is listed between members of the IX. These IX route servers will not show up in the simulation; for reference they are listed in table 4.2. Fortunately these situations with private peers and IX route servers do not affect most of the research on attack and defense. This is due to the fact that peering creates isolated islands that do not propagate announcements.

In any event, the simulation model is designed to be basically simple and reasonably accurate without adding undue complexity. It cannot reflect the individual policy maintained by each router in each AS as most of these are not known. Instead, the simulator uses the valley-free rule as its generic propagation policy. Egress filters that prevent improper BGP announcements from propagating are not initially enabled; egress filters will be applied when we discuss attacks and defenses. Furthermore, the simulation does not account for policy unique to each geographically

Table 4.2. *Internet Exchange List*

ASN	Internet Exchange Name
1200	AMS-IX1
4635	HKIX-RS1
5507	Budapest Internet Exchange
6695	DECIX-AS
7606	Western Australia Internet Exchange (WAIX)
8714	London Internet Exchange
9355	Nat'l Institute of Information & Communication Technology
9560	Auckland Peering Exchange (APE)
9439	Wellington IX (WIX)
9722	PIPE
9989	EQUINIX-AP
11670	AS-TORIX
17819	ASN-EQUINIX-AP
18398	PIPE Networks Sydney Internet Exchange
21371	EQUINIX-UK-ASN
24029	NIXI is an IXP in India
24115	ASN-EQIX-MLPE
24990	EQUINIX-FR-ASN
35054	EQUINIX-CH-ASN
40633	Los Angeles Internet Exchange
42476	Swiss Internet Exchange SwissIX
43100	Lyonix
47886	EQUINIX-NL-ASN
48850	KIX - KIKE Internet eXchange
55818	MC-IX

diverse POP that an AS manages. It does not inflate paths by repeating AS numbers. It does not dynamically announce and withdraw paths to perform traffic management.

There is ongoing research to improve AS relationship inference. A very recent Infocom paper by Neudorfer *et al* [33] describes improvements using Point-of-Presence (PoP) information. (Coincidentally, this paper also noticed the Stanford anomaly that I had discovered.) But perfection is not the goal. Despite its limitations, the simulation model is reasonably accurate and it provides an extremely useful tool for studying complex BGP interactions.

As one final sanity check, I tried duplicating a real world IP Hijack attack with the simulator. ArsTechnica and several blogs [39] had reported an event on November 5, 2012. An Indonesian ISP, Moratel, inadvertently hijacked Google for 30 minutes by announcing the Google address space. Only a small percent of the internet was affected, but the report indicated that the CloudFlare AS was affected. My simulation reported that 9% of the internet address space was affected, and the simulated AS for CloudFlare was impacted as well. Although this experiment is more descriptive than analytical, it still bolstered confidence in the simulation accuracy.

4.3. *System Measurements*

The AS system topology is the stage upon which all BGP announcements, attacks and defenses are performed. This section provides basic measurements such as size and connectivity to help in understanding BGP behavior. We also introduce several new metrics, topological *depth* distribution, *tier-1 reach* and *tier-1 overlap* that correlate with attack propagation. The usefulness of these new metrics will be seen in subsequent chapters.

Table 4.3 presents basic measurements regarding the size of the AS topology. There are a total of 42,697 ASes in the system. From top to bottom of the hierarchy, these consist of:

- **tier-1**: 17 of the ASes qualify as tier-1 ASes. These nodes are fully interconnected; that is, each tier-1 AS is directly connected to all other tier-1 ASes. By definition, a tier-1 AS has no provider relationships; their role is to provide transit for their peers and customers. The largest tier-1 AS is Cogent, with 3,738 connections to neighboring peers and customers. The smallest is France Telecom, with 142 neighbors. A list of tier-1 ASes is provided in table 4.4.

Table 4.3. *Topology Characteristics*

Category	**Count**
Total number of ASes in network	42,697
Tier 1 Transit ASes	17
Non-Tier1 Transit ASes	6,301
Stub ASes	36,379
Stub ASes connected to a Tier-1	10,725
Stub ASes singly-homed to Tier 1 ASes	2,418
Transit ASes homed to Tier 1 ASes only	3,080
Highest degree of inter-connections	3,738
Maximum Distance from a tier-1 AS	7

- **Transit**: 6,301 (14.7%) transit ASes exist at one or more levels below the tier-1 ASes. A Transit AS can have zero or more providers, peers, customers and siblings. Approximately half (3,080) of the Transit ASes are directly connected to a tier-1 AS. The remainder are lower in the hierarchy and connect to a higher-level transit-AS.
- **Stub**: 36,370 (85%) of the ASes are stubs; that is, they have no customers and are connected to providers, peers or siblings only. Of these, 10,725 stubs are directly connected to a tier-1 AS, and 2,418 of these are singly-homed. The maximum distance from a tier-1 AS to a Stub AS is 7 hops.

The next set of measurements addresses inter-connectivity among the ASes. Figure 4.7 charts the distribution of ASes with respect to *degree* (the total number of neighbors connected to an AS), and then further decomposes this into *provider, customer and peer* distributions. The data show that an AS may have as few as 1 neighbor and as many as 3738. As is evident from the chart, however, about 80% of the ASes are connected to just 3 or fewer neighbors. Classifying these connections we observe:

- **provider**: about 80% of the ASes have 0-2 providers, with the maximum at 37.

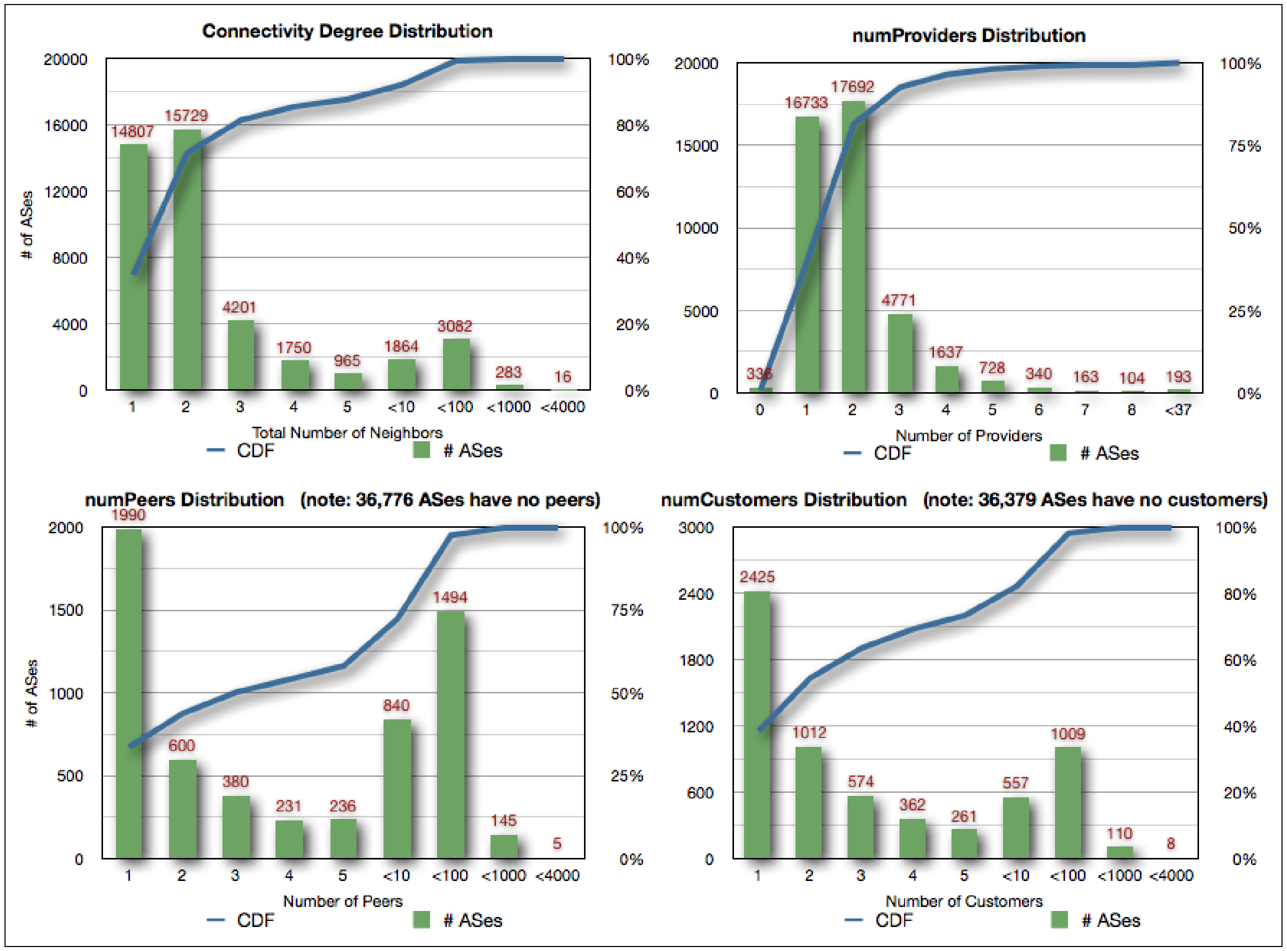


Figure 4.7. *Base Metrics: distribution of degree, providers, peers and customers*

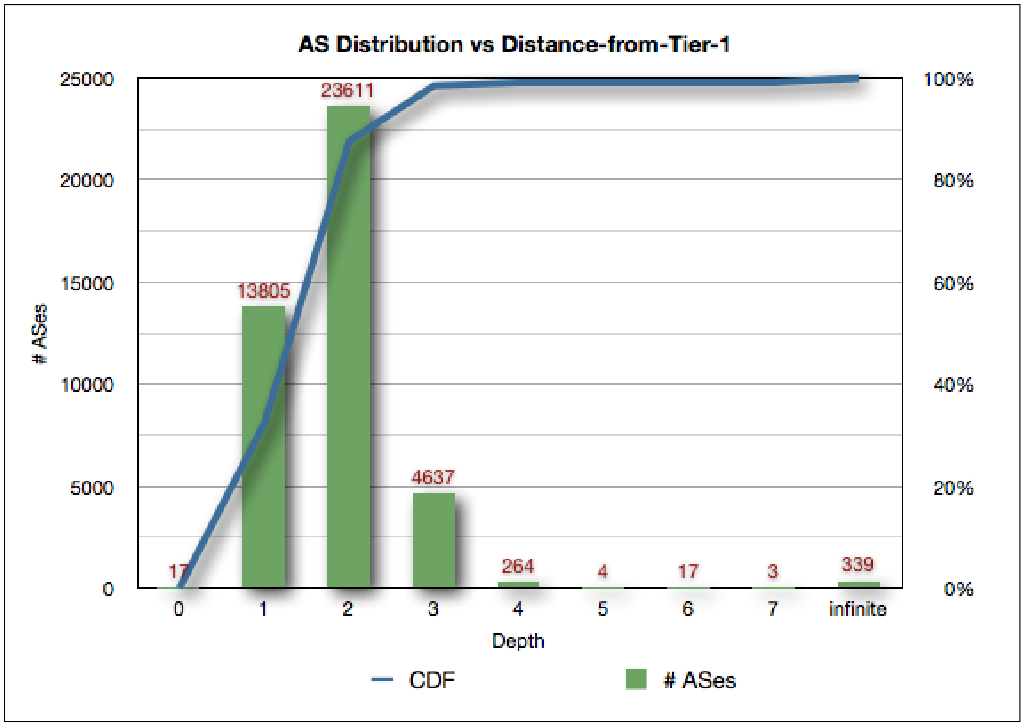


Figure 4.8. Depth *distribution*

- **customer**: Most of the ASes are stubs; 36,379 of the ASes have no customer links. Of the remaining 6,318 ASes, 80% have 10 or fewer customers and 99.7% have fewer than 100 customers. The largest number is 3,659.
- **peer**: 36,776 of the ASes have no peer relationships. The remaining 5,921 ASes have a slightly broader distribution of peers; 72% have fewer than 10 peers. The maximum is 2,029. It should be noted that *internet exchange points* are centers of connectivity in which hundreds of ASes can peer with each other. Examples include Equinix with 768 members, DE-CIX (Deutscher Commercial Internet Exchange) with 480+ members, PARIX (Paris Exchange) with 40+ members, and dozens more.

Next, we define several new metrics that play prominently in the success or failure of attack propagation, *depth*, *tier-1 reach*, and *tier-1 overlap*.

Depth is defined as the distance from an AS to its nearest tier-1 AS. By definition, this number will be 0 for a tier-1 AS, and infinite if there is no path from an AS to any tier-1 AS. The maximum depth is seven, however 99% of ASes have a depth less than or equal to three as illustrated in Figure 4.8. Depth is important because it relates to BGP path length. As described in Chapter 2, BGP path length is a major factor in accepting or rejecting a route. An attacker can be more successful if its depth is low resulting in shorter attack path lengths. Conversely, a target becomes more vulnerable with higher depth because of the increase in path length.

Tier-1 Reach is defined the total number of ASes that a tier-1 BGP announcement will reach if all other tier-1 ASes are disabled. This gives a measure of BGP propagation as well as a measure of independence and overlap among the various tier-1 ASes. *Reach* ranges from a minimum of 21,468 to a maximum of 42,321. Table 4.4 lists the seventeen tier-1 ASes and provides several metrics including *reach* and *degree*.

Unlike other metrics such as *AS degree*, it should be noted that *Tier-1 reach* is not a measurement that can be simply calculated or looked up. *AS reach* can only be derived through simulations that propagate announcements through a topological model.

It is interesting to note that there is little correlation between the *degree* of a tier-1 AS and its corresponding *reach*. For example, ATT has a relatively large *degree* of 2430 and can propagate an announcement to 21,297 ASes. On the other hand, France Telecom has a very small *degree* of 142, yet it can reach 21,468 other ASes, slightly more than ATT. How can this be explained?

An examination of the simulation results announcing ATT and France Telecom shows that, as expected, the first generation of BGP announcements propagates to 2430 and 142 ASes respectively. However the second through ninth generations continue the propagation and it is the AS-connectivity among these levels that determines the final outcome.

Table 4.4. *Tier-1 AS List*

ASN	Name	Degree	Peers	Customers	Reach	Reach2	Overlap
174	Cogent	3738	80	3,658	32,172	35,980	89%
3356	Level3	3524	56	3468	35,477	37,006	96%
7018	ATT	2430	31	2399	21,297	33,378	64%
3549	GBLX	2376	433	1943	35,736	36,828	97%
701	UUNET	1765	44	1721	23,143	32,801	71%
209	Qwest	1433	54	1379	28,460	33,488	85%
2828	XO	1173	151	1022	29,500	33,798	87%
6461	Metromedia	1067	166	901	32,040	34,600	93%
3257	TINET	968	59	909	32,369	35,065	92%
1239	Sprint	929	35	894	27,520	33,048	83%
2914	NTT	882	78	804	31,732	34,637	92%
1299	TeliaNet	800	41	759	32,969	35,438	93%
6453	TATA	712	109	603	34,619	35,890	96%
3320	DTAG	547	70	477	24,848	32,278	77%
3561	Savvis	386	41	345	34,856	35,929	97%
6762	Seabone	304	60	244	28,004	33,177	84%
5511	France T'com	142	33	109	21,468	31,435	68%

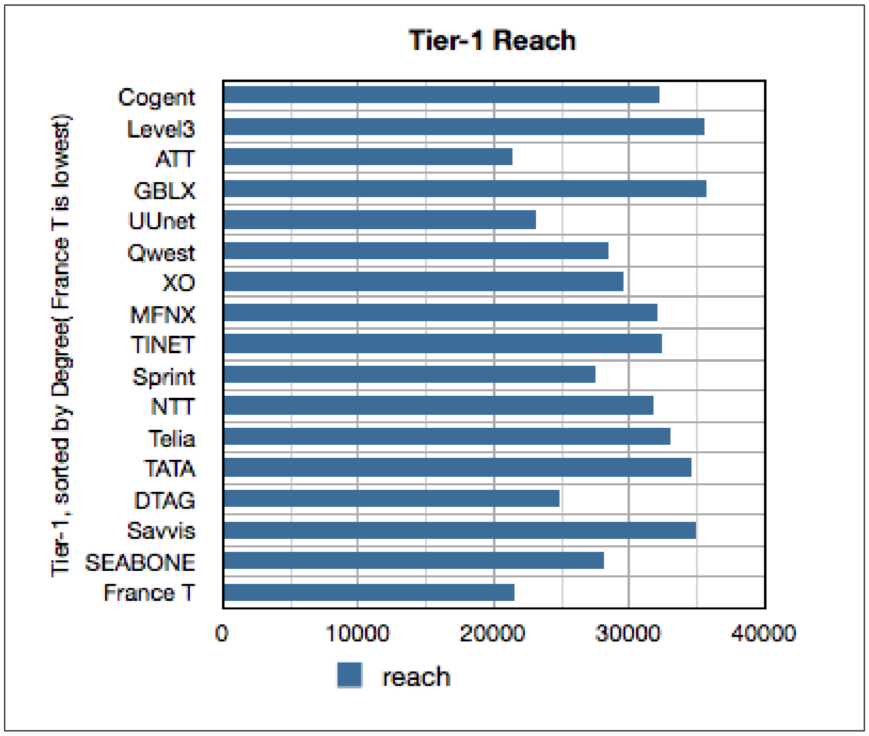


Figure 4.9. *Tier-1 Reach (sorted with respect to* degree*)*

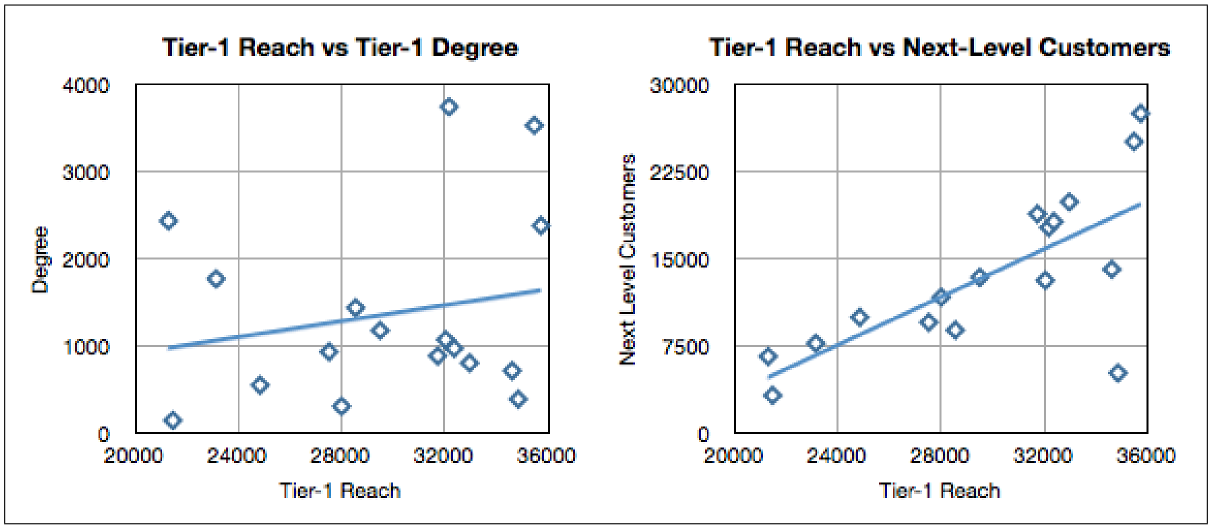


Figure 4.10. *Scattergrams: Tier-1 Reach vs Degree (correlation=0.20) and Tier-1 Reach vs Next-Level-Customer-Count (correlation=0.72)*

The scattergrams in Figure 4.10 were created in an effort to corroborate the results seen during the simulation. The first graph shows *reach* vs *degree* and indicates a very low correlation of 0.20. (A correlation greater than 0.8 is generally described as strong, and anything less than 0.5 is considered weak.) The second scattergram shows *reach* vs *sum of 1st level customer count* and shows a better correlation of 0.72. Accordingly, it is shown that each tier-1 AS can propagate quite differently from other tier-1 ASes based not on *degree* but on the underlying mesh of connectivity.

The *reach* metrics can also be used to measure the relative independence of each tier-1 AS. For example, how much overlap is there between announcements sent from ATT and from Level3? Likewise, how independent are they?

To answer this, Figure 4.11 shows 4 of the 17 possible charts when announcements are made from pairs of Tier-1 ASes.

France Telecom is shown to have a small reach of 21,468 on its own. When combined with any other Tier-1 AS, their combined reach is extended to various higher levels, averaging to 31,435.

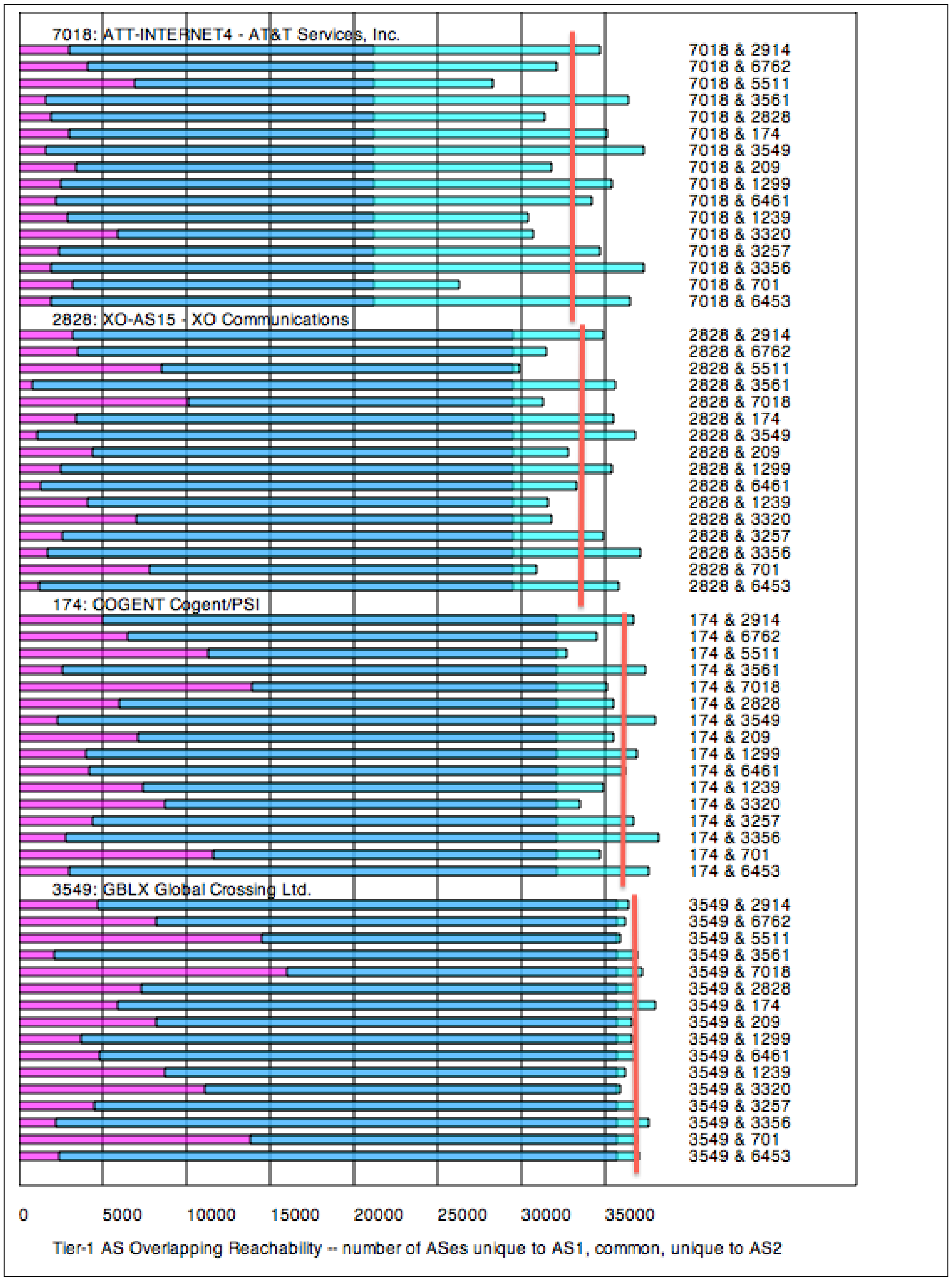


Figure 4.11. *Four set of Tier-1* overlap *measurements (red bar = average)*

The *overlap* metric is calculated as *dualReach / standaloneReach*. France Telecom's *overlap* metric calculates to 68%. In other words, France Telecom has a comparatively low overlap with other Tier-1 ASes.

On the other hand, Global Crossing has a large standalone reach of 35,736. When combined with another Tier-1 AS the average reach only extends a small amount to 36,828. The *overlap* metric in this case is 97%; that is, Global Crossing greatly overlaps with the other Tier-1 ASes.

Figure 4.12 summarizes all of the Tier-1 ASes listing individual reach and average overlapping reach. This table of *overlap* metrics gives an indication of BGP behavior when an AS is multi-homed with two or more Tier-1 providers.

Now that we have a set of basic metrics, we can proceed with an analysis of IP hijack propagation. This will be discussed in the next chapter.

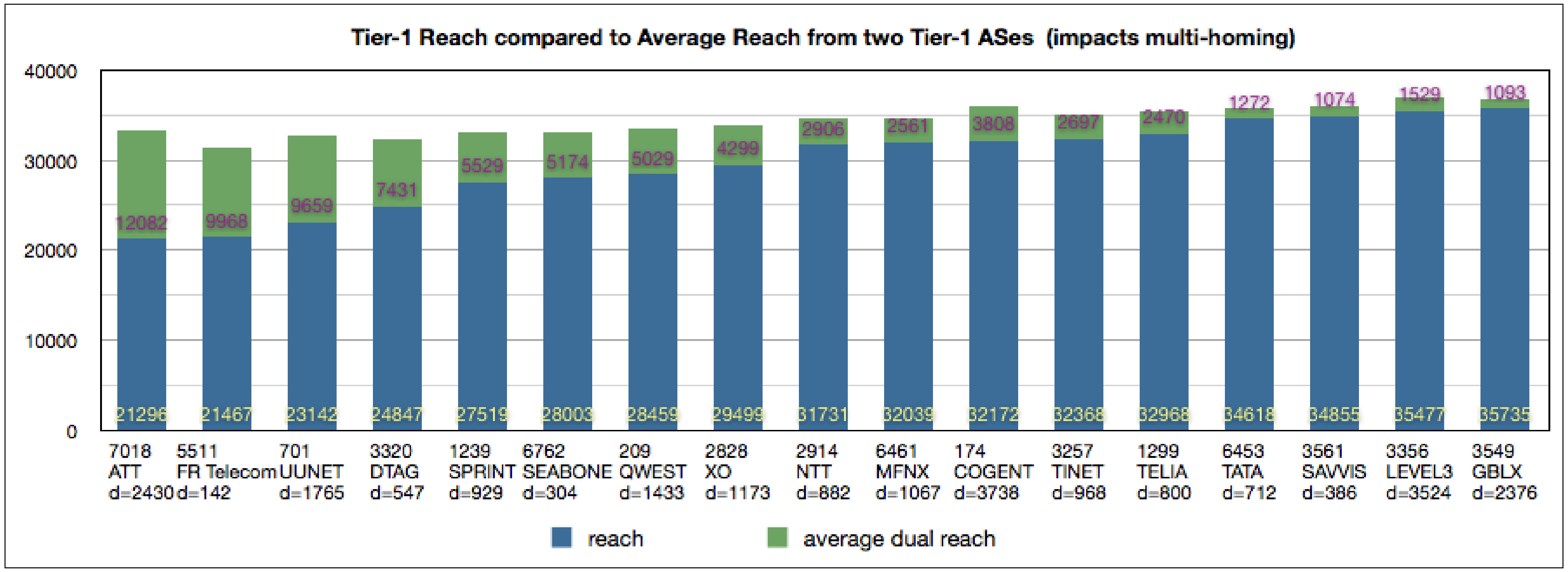


Figure 4.12. *Tier-1* reach *and* overlap

CHAPTER 5

ANALYSIS OF IP HIJACK PROPAGATION

Before we can analyze the effectiveness of ROVER, we must first understand how IP hijacks propagate in the absence of any filters or blocking devices. This provides a baseline for comparisons when we do apply defensive measures.

We define a target AS as *vulnerable* if a large number of other ASes are polluted with a bogus path when an attacking AS makes an announcement hijacking the target's address space. Conversely, a target is *resistant* when relatively fewer ASes are polluted. Likewise, an attacker is considered to be *aggressive* if it can pollute many ASes compared to the average case.

Attack propagation can vary immensely. Sub-prefix attacks are the most aggressive. Since they have no competing path to overcome, sub-prefix attacks are able to propagate across and pollute the entire Internet. In contrast, origin attacks can be either strong or weak. Some ASes are more or less vulnerable to attack than others. Some ASes are more or less aggressive as an attacker. When an origin attack does occur, it could end up polluting just a portion of the Internet, or spread very widely.

In this chapter we examine these variations in attack propagation and analyze reasons for the different behavior. Section 5.1 demonstrates attack variance with two illustrative examples. Sections 5.2 and 5.3 gives an analysis of the topological factors that affect the extent of an attack propagation. Section 5.4 then gives a brief overview of the changing Internet topology with respect to private peering and Internet Exchanges and how this may affect attack propagation.

5.1. *Attack Variance*

To illustrate attack variance, we will describe two origin hijacks that have very large differences in their attack extent. The first example is a small attack in which an Indonesian ISP, Moratel, at AS 23947 attacks Google at AS 15169. The second is a much larger attack in which ISI at AS 4 attacks Harvest Electronics NZ at AS23947. These attacks are depicted in figures 5.2 and 5.3.

Please note that we have slightly modified the format of these polar graphs (see Figure 5.1. The new graphs are now constructed such that the *depth2* metric for the AS is plotted along the radius rather than *degree*. The *Depth2* metric is similar to *depth*, but allows both Tier-1 and large Tier-2 backbone networks such as Time Warner Cable to also have a value of zero. The rationale for this will be explained in Section 5.2.

Depth2 ranges from 0 to a maximum of 5, and one can see the number of ASes decreasing in each band, as would be expected. The final circle in the middle is "infinity", for the few ASes for which no path exists from a Tier-1 or Tier-2 provider to the AS. This scheme creates seven concentric circles, one for each value of depth, with highest depth in the center of the graph. This new format makes it easier to see the communication between neighboring bands. AS *degree* is still shown, but is now indicated by scattering within a concentric circle. Higher degree ASes are towards the center. A sample of the new format polar graphs is shown in figure .

Figure 5.2 shows a sequence of events as Moratel hijacks Google's address space. At the end of the attack, only 3,416 of the total 42,697 ASes become polluted (18% of the internet address space). Figure 5.3 shows ISI attacking Harvest Electornics New Zealand. In this case 40,950 ASes become polluted (96% of the address space).

These two cases display a *huge* difference in behavior. In both cases we see the small number of messages sent from the attacker to its neighbors in the first generation of announcements. The

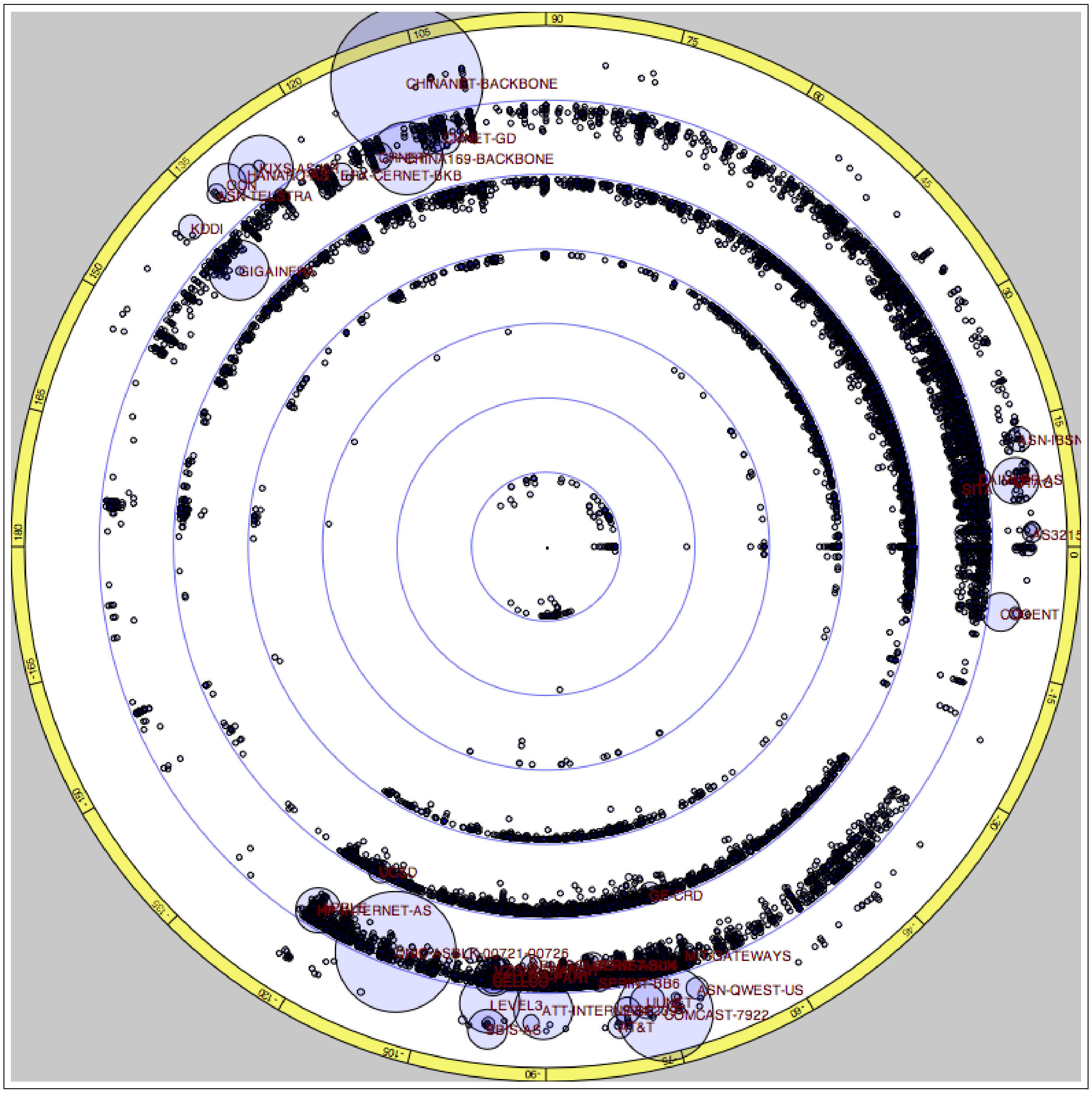


Figure 5.1. *Polar graph of Network Topology where radius is now determined by* depth

remaining three graphs in each sequence show subsequent generations of BGP message propagation. Red lines indicate a BGP announcement that is accepted by an AS, polluting its RIB with a bogus path. Green lines indicate a BGP announcement that is rejected by the AS because it already has an equal or better path than the one being received.

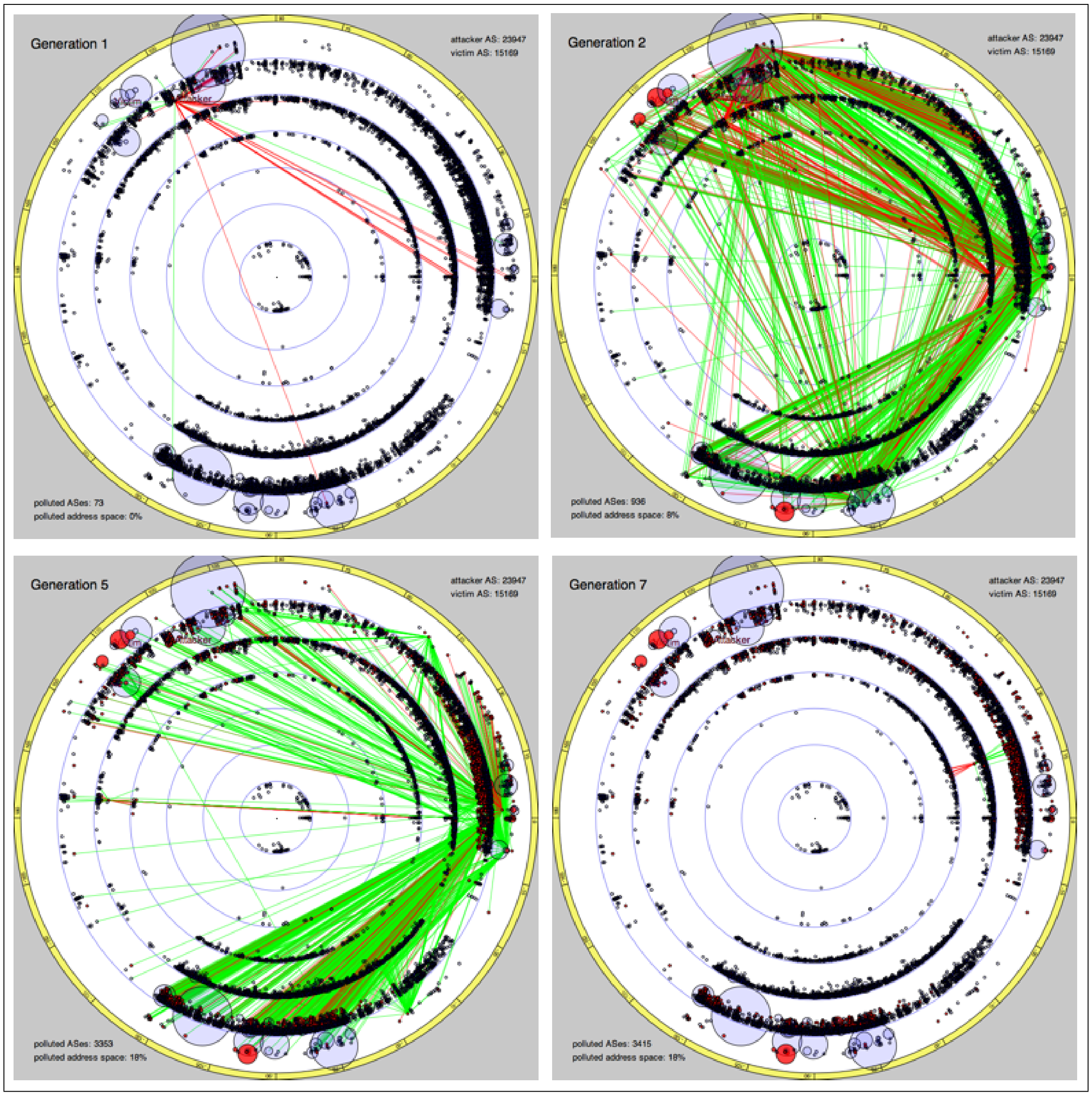


Figure 5.2. *A small attack: AS23947 (Moratel) hijacking AS15169 (Google)*

In the Google example, most of the lines are green, and the attack doesn't get very far. In the Harvest Electronics case, however, the attack begins to succeed in the second generation and by the third generation there is a veritable storm of bogus announcements being accepted as the attack cascades into a major incident. The reason for the large difference in behaviors between the two

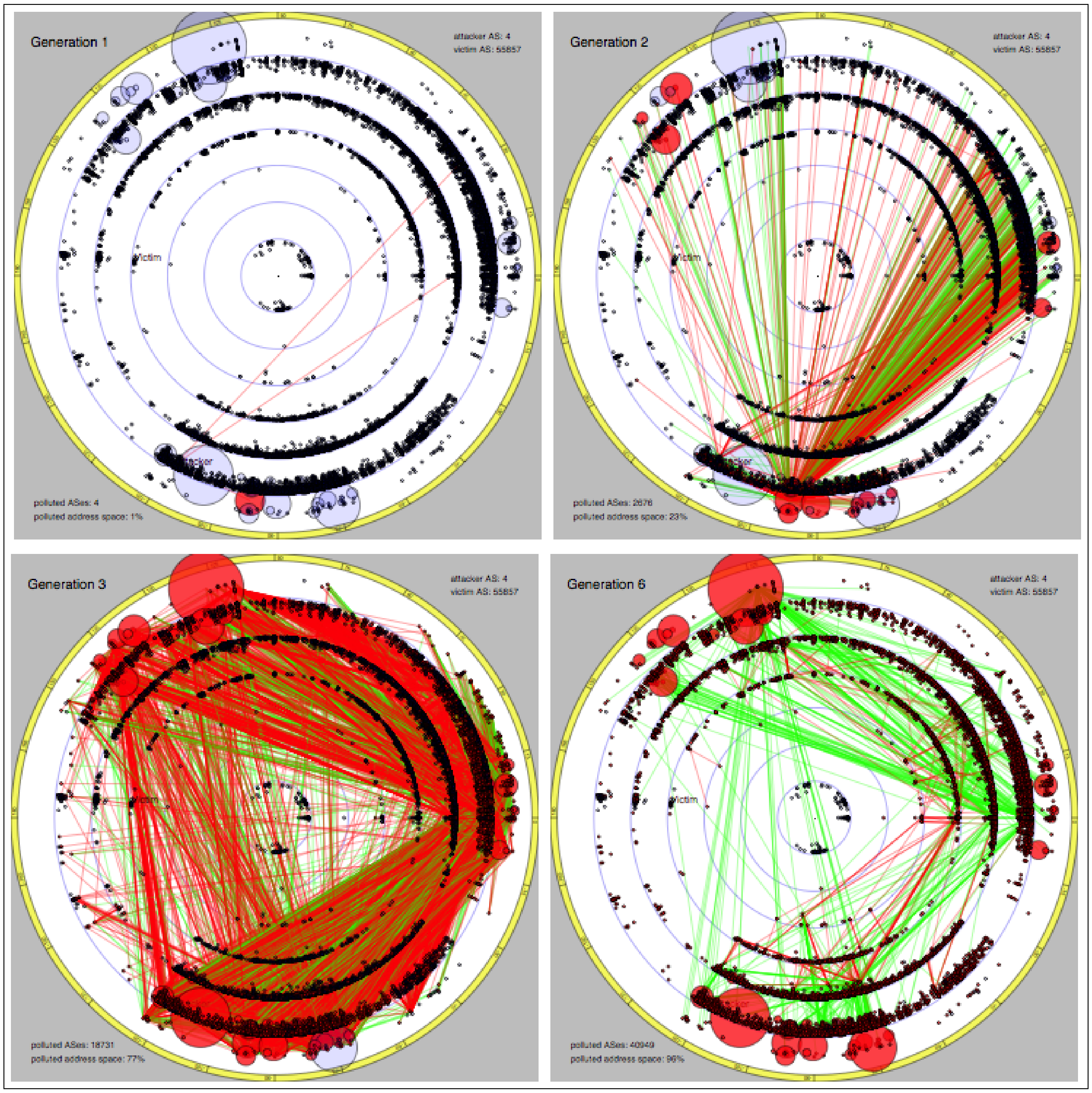


Figure 5.3. *A much larger attack: AS4 (ISI University of Southern California) hijacking AS55857 (Harvest Electronics NZ)*

attacks is related to the path lengths associated with each AS. This will be explained in Section 5.2.

Another experiment was conducted to examine variances. In this experiment, separate attacks were launched from each and every AS against a single target, AS25883 (Citigroup). Even when

Table 5.1. *Variance in Attack Propagation for Individual ASes Attacking AS25883 (Citigroup)*

Attack Extent (Polluted ASes Created)	Number of attacking ASes having this extent
0	580
1 to 10	246
10 to 100	571
100 to 1000	4646
1000 to 10000	35225
10000 to 42697	1429

the target AS remains the same, the extent of the damage varied widely depending on which AS performed the attack (see table table 5.1).

5.2. *Topological Factors Affecting Attack Propagation*

Why is it that some attacks propagate widely while others are hardly noticeable? Is it possible to predict, or at least give some general observations that quantify the extent of an attack?

This question was first examined by Ballani *et al*[1]. The Ballani paper, however, focused not on quantification, but on the success or failure of an attack based on two variables, *AS relationship* and *path length*. In this chapter we augment [1] by analyzing case studies in which the topological locations of the attacker and the target are varied and measure the extent of each attack using the simulator.

In general, predicting the extent of an attack can be very difficult. The topological path between each AS in the system with respect to an attacker and a target can vary widely, causing large divergence in the attack propagation. It is obviously impractical to consider each of the 1.8 billion combinations of attackers and targets to build an exhaustive (and exhausting) quantification study. Therefore the approach will be to search for patterns within the topology that exhibit a a statistical consistency to their behavior. In the worst case, when no such pattern can be found, we can always fall back to the simulator to get a numerical quantification of an attack extent.

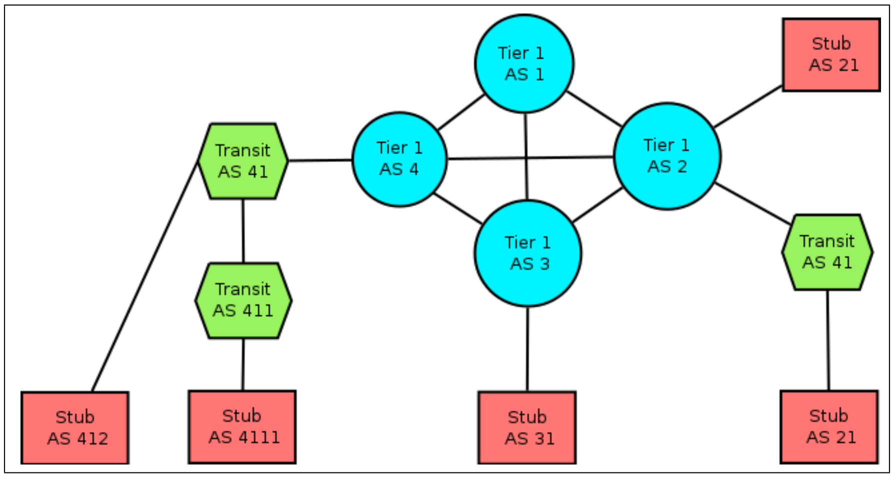


Figure 5.4. *Simplified AS topology*

Table 5.2. *Summary of Average Attack Magnitudes; SH=singly homed, MH=multiple-homed, δ specifies AS depth*

Δ_{depth}	**Attacker**	**Target**	**Polluted ASes**	**Average**
-1	Stub, SH, δ1	Tier1, δ0	0	0
-1	Stub, SH, δ2	Stub, SH, δ1	1 - 40	9
0	Tier1, δ0	Tier1, δ0	513 - 19,032	6,040
0	Stub, SH, δ1	Stub, SH, δ1	513 - 19,032	6,040
0	Stub, SH, δ1	Stub, MH, δ1	0 - 15,594	3,714
0	Stub, MH, δ1	Stub, SH, δ1	2,691 - 25,059	10,187
0	Stub, MH, δ1	Stub, MH, δ1	544 - 21,235	7,286
0	Transit, δ1	Transit, δ1	1,249 - 17,864	6,467
0	Stub, SH, δ2	Stub, SH, δ2	1,249 - 17,864	6467
1	Tier1, δ0	Stub, SH, δ1	22,053 - 41,798	34,337
1	Tier1, δ0	Stub, MH, δ1	16,102 - 40,287	29,462
1	Stub, SH, δ1	Stub, SH, δ2	22,053 - 41,795	33,760
1	Transit, δ1	Stub, SH, δ2	22,746 - 41,798	34,241
2	Tier1, δ0	Stub, SH, δ2	30,486 - 41,913	40,016

To aid in this discussion, Figure 5.4 serves as a simplified topology guide. The Tier-1 ASes are shown in blue, fully inter-connected with each other. Stub ASes (red) are shown at depth 1, and at depth 2 and 3 as they connected through transit ASes (green). There are obviously other more complicated situations, but this provides a starting guide. Later we will expand the discussion to include major Tier-2 backbones, but for now we are examining Tier-1 centric networks.

Table 5.3. *Attack Magnitude as a Matrix*

		Target				
		Tier1 $\delta 0$	Stub, SH $\delta 1$	Stub, MH $\delta 1$	Transit $\delta 1$	Stub, SH $\delta 2$
Attacker	Tier1, $\delta 0$	6,040	34,337	29,462		40,016
	Stub, SH, $\delta 1$	0	6,040	3,714		33,760
	Stub, MH, $\delta 1$		10,187	7,286		
	Transit, $\delta 1$				6,467	34,241
	Stub, SH, $\delta 2$		9			6,467

Many experiments were conducted by attacking stub-to-stub, stub-to-Tier-1, etc. Results from these experiments are shown in tables 5.2 and 5.3 and summarized in the following list:

(1) **Factors affecting Attack Magnitude**

- Define $\Delta_{depth} = depth_{defender} - depth_{attacker}$
- Δ_{depth} is the primary factor dictating the magnitude of an attack. An examination of table 5.2 shows a marked increase in average AS pollution for each change in Δ_{depth}.
- Sub-factors affecting attack magnitude are *Tier-1 reach* and *Tier-1 overlap*. The range of *min* to *max* strongly correlates with *Tier-1 reach*.
- *AS degree*, although a well known metric, shows little correlation to attack magnitude.

(2) **Effect of Multi-homing and AS type**

- Multi-homing makes an attacker more aggressive.
- Likewise, multi-homing makes a target less vulnerable.
- AS type (Tier1, stub, transit) appear to have little affect on attack magnitude, although they obviously play a part in the policy decisions for attack propagation.

(3) **Attacker Aggressiveness** Since Δ_{depth} is the primary predictor, an aggressive attacker should be a Tier-1 AS at depth 1. Multi-homing will increase aggressiveness. Carefully selecting the provider ASes to have maximum *AS reach* and minimumAS overlap will increase aggressiveness.

(4) **Target Vulnerability** The same factors listed above that make an attacker more aggressive can also be used to make a target less vulnerable.

To understand the differences obtained in each experiment, recall the two primary factors used by each AS to independently accept or reject an incoming announcement: LOCAL_PREF and PATH LENGTH.

- If a non-Tier-1 AS receives an announcement carrying a higher LOCAL_PREF than is currently stored in the RIB, it will accept the new announcement and propagate it to its appropriate neighbors. (This does not apply to Tier-1 ASes which use PATH LENGTH only).
- If an AS (both Tier-1 and non-Tier-1) has the same LOCAL_PREF but a shorter PATH LENGTH, the AS will accept the announcement and propagate it to its appropriate neighbors, bogus or not.
- Otherwise the announcement is rejected.

A particular attack can be analyzed by looking at the topological location of each AS and following the rules just listed: at each step, analyze whether the attack will propagate *up* (to providers), *down* (to customers), and/or *across* (to peers and the customers of those peers. Each AS must compare the path length to the attacker vs the existing path length to the target. If the attacker has a shorter path length, the attack will be accepted and continue to propagate.

5.2.1. *Attacks at* $\Delta_{depth} = 0$. In this section we examine the results when a Tier-1 AS attacks another Tier-1 AS. The analysis and measurements are similar for other *depth-n* vs *depth-n* attacks, although there will be intervening transit ASes involved that can result in slightly higher pollution.

A Tier-1 AS, due to its position in the core of the AS topology, is a very aggressive attacker. The number of ASes polluted when one Tier-1 AS attacks another Tier-1 ranges from a minimum of 513 (France Telecom attacking Global Crossing) to a maximum of 19,032 (Global Crossing attacking France Telecom). The average value is 6,040.

This average number is fairly low, considering that Tier-1 ASes are strong attackers. Counter-balancing this, however, is the fact that Tier-1 ASes are not very vulnerable as targets.

Tier-1 vs Tier-1 attacks are reasonably simple to analyze. An "attack" announcement cannot propagate *up* since a Tier-1 has, by definition, no providers. Nor can the announcement propagate *across* to other peer Tier-1 ASes. That attack announcement will have a path length of 1, which is equal to the length of the pre-existing valid announcement, so it will be rejected. The only direction in which the announcement can propagate is *down* to the attacking Tier-1 AS's own customers.

Figures 5.5 illustrates the average number of ASes polluted for each Tier-1 AS acting as either an attacker or as a defender. The ASes are sorted by *AS degree* and corresponding correlation charts are also shown. Although *AS degree* is a well-known metric, *AS degree* fails to show a very high correlation as a predictor of attack extent (r values = -0.34 and 0.69 respectively). In contrast to this, the new *AS reach* metric introduced in the previous chapter actually shows a stronger correlation of -0.88 for the target and approximately the same correlation for attacker.. Figure 5.6 presents the data sorted by *AS reach* along with associated correlation charts. A quick scan of these charts reveals two outliers: Savvis and France Telecom. It is likely that the very low *AS degree* of these two ASes dominates the *AS reach* metric, unlike the remaining ASes with moderate to high

AS degree values. If these two outliers are removed from the chart, correlation increases to -0.94 (very strong correlation) and 0.76 (high-moderate correlation).

A hypothesis that multiplying *AS degree* with *AS reach* could be a better predictor of attack extent was proposed, but the resulting correlation numbers did not support this experiment. Future work could examine all 289 unique cases rather than the averages, and could also examine the overlap factors. But for now, *AS reach* appears to be the major predictor quantifying attack extent. The total polluted ASes is proportional to but less than *AS reach* due to the interference from overlaps with other Tier-1 ASes.

Finally, we should note that other *depth-n* vs *depth-n* attacks do propagate BGP announcements UP and back DOWN again. The intervening transit ASes may have descendants that become polluted, which will increase the total aggregate pollution. This is observed in table 5.2.

5.2.2. *Attacks at* $\Delta_{depth} = 1$. We next examine what happens when a "blue" Tier-1 AS in Figure 5.4 attacks a singly-homed "red" Stub AS directly connected to a Tier-1 AS. This is an example of a $\Delta_{depth} = 1$ attack.

The number of ASes polluted in this type of attack ranges from a minimum of 22,053 (France Telecom attacking any stub singly-homed to Level3) to a maximum of 41,798 (NTT attacking any of its own stubs). Otherwise, the maximum for a Tier-1 attacking someone else's stub was 40,982 (TINET attacking any stub of France Telecom). The average value is 34,377.

This average number is much higher than before, as Δ_{depth} increases from 0 to 1.

Tier-1 vs Stub-1 attacks are also simple to analyze. As before, an "attack" announcement cannot propagate *up* since a Tier-1 has, by definition, no providers. In this case, however, the attack does propagate *across* to peers replacing the valid announcement with path length of 2 with a new bogus announcement with a shorter path-length of 1. All of the tier-1's except for the one

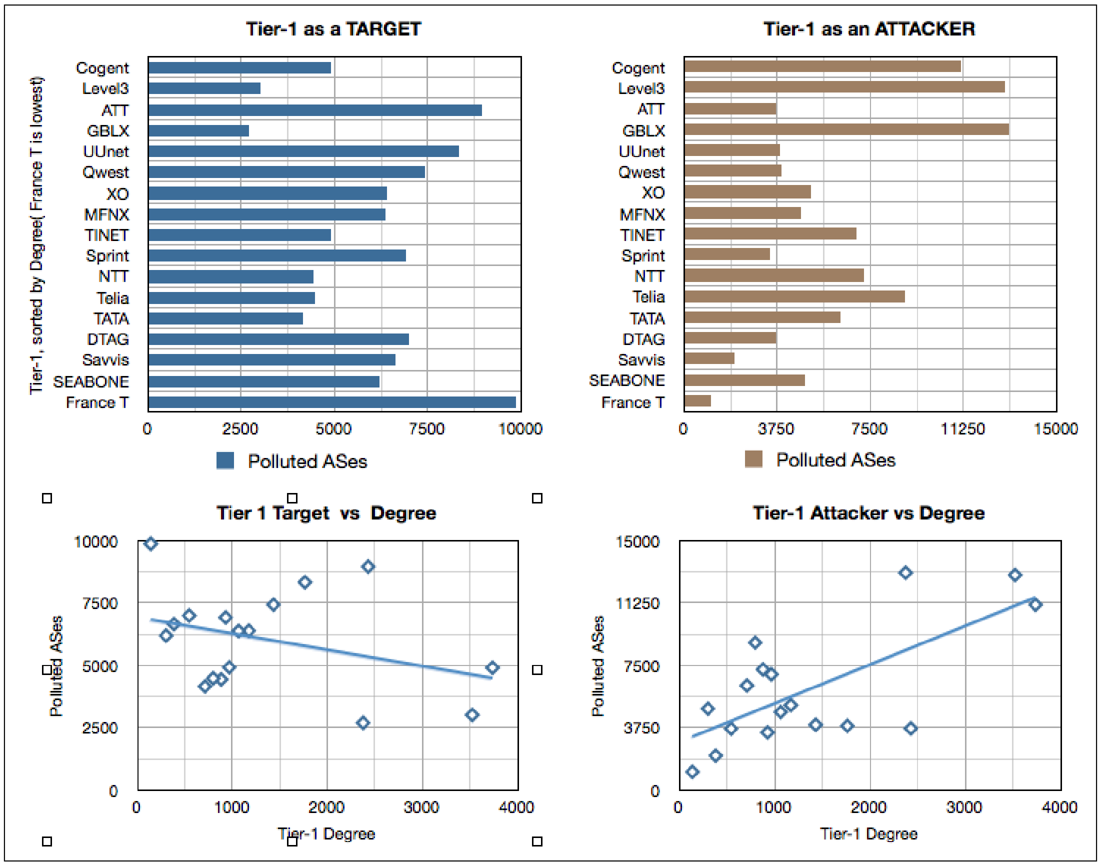


Figure 5.5. *Tier-1 Analysis with respect to* degree*; TARGET correlation=-0.34 (weak); ATTACKER correlation=0.69 (medium)*

directly connected to the stub become polluted and participate in propagating the attack to their descendants. With this many Tier-1 participants it is easy to see why we get the massive numbers for the extent of the pollution.

In this scenario, it makes more sense to ignore the large number of polluted ASes and focus instead on the much smaller number of non-polluted ASes. Why do they remain intact? The reason is that there is only one non-polluted Tier-1 AS: the one connected to the stub itself. This Tier-1 will not pollute its descendants. However some of these descendants can be polluted if they happen to overlap and have short enough connectivity to the other Tier-1 ASes. Consequently, the number of non-polluted ASes will be less than or equal to the *AS reach* of that unpolluted Tier-1.

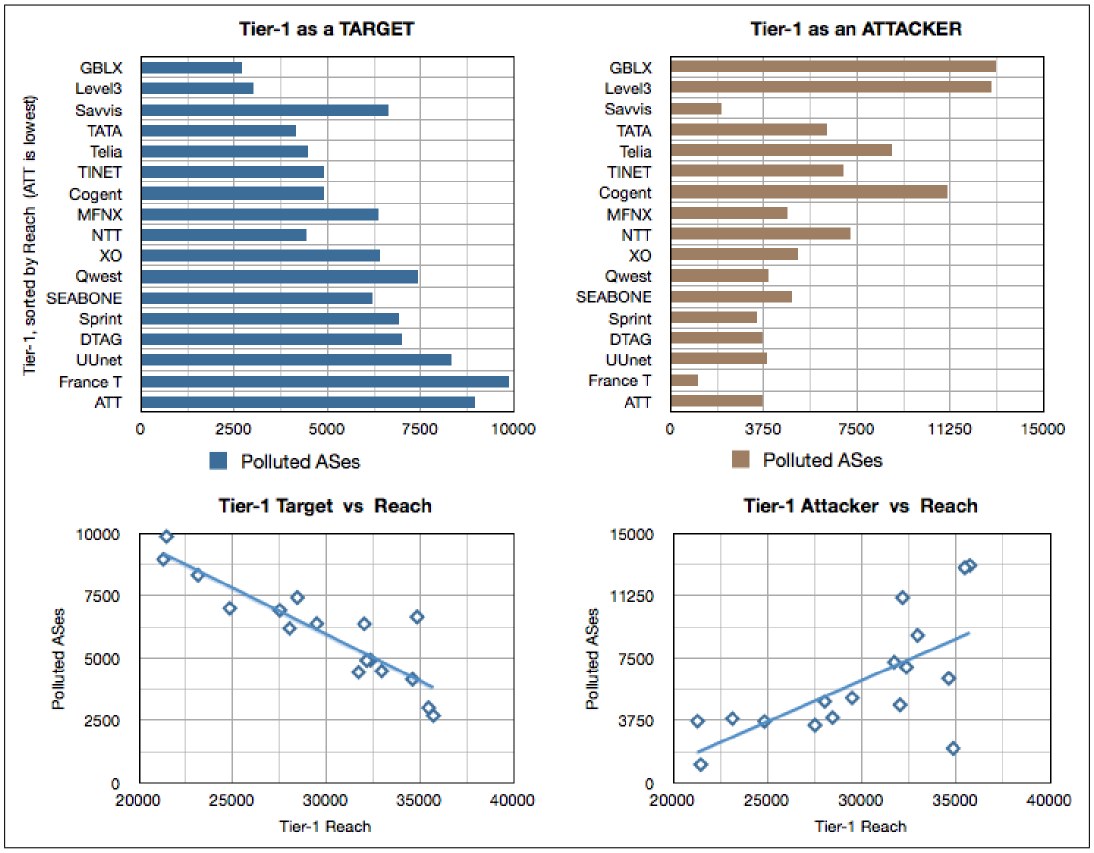


Figure 5.6. *Tier-1 Analysis with respect to* reach*; TARGET correlation=-0.89 (strong); ATTACKER correlation=0.66 (medium). Removing Savvis and FT outliers increases correlation to -0.94 and 0.76 respectively.*

Figure 5.7 shows the data correlates reasonably well with *AS reach*. Attacking Tier-1's have a positive correlation of 0 .77, and the defending stubs have a negative correlation of -0.88. The value is negative because the higher the reach, the fewer the number of polluted ASes.

5.2.3. *Attacks at* $\Delta_{depth} = 2$. Deeper ASes become more difficult to analyze due to variances in the intervening transit ASes. However we can expect ASes at deeper levels will be more vulnerable as Δ_{depth} increases.

Supporting this argument is the experimental data in which a Tier-1 attacks a Stub at depth 2. In this case the average number of polluted ASes is 40,016. This is about 5,000 more than the previous case at depth 1. The data is illustrated in Figure 5.8.

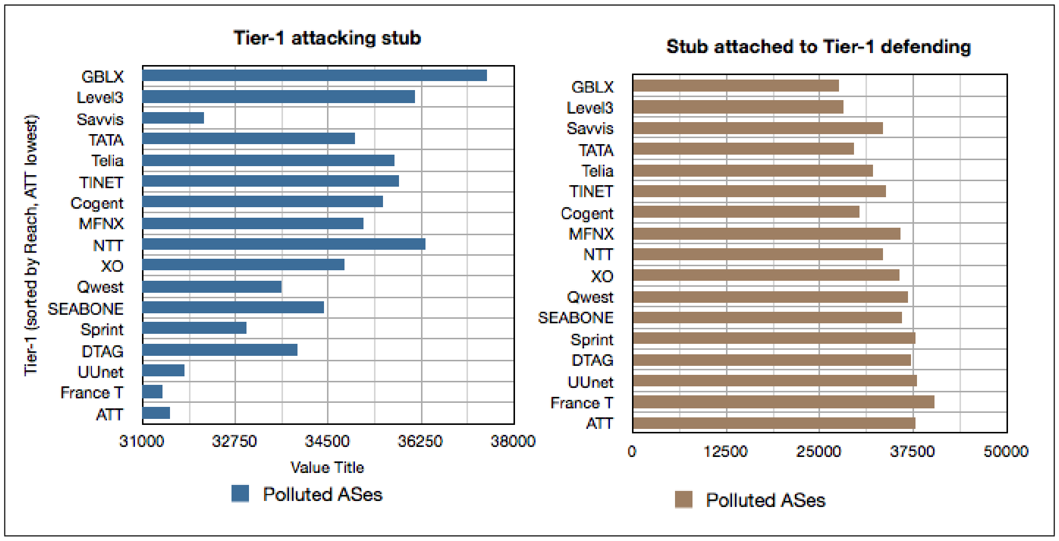


Figure 5.7. *Analysis of Tier1 attacking a Stub at depth 1 with respect to* reach*; ATTACKER correlation=0.77 (high-medium); DEFENDER correlation=-0.88 (strong)*

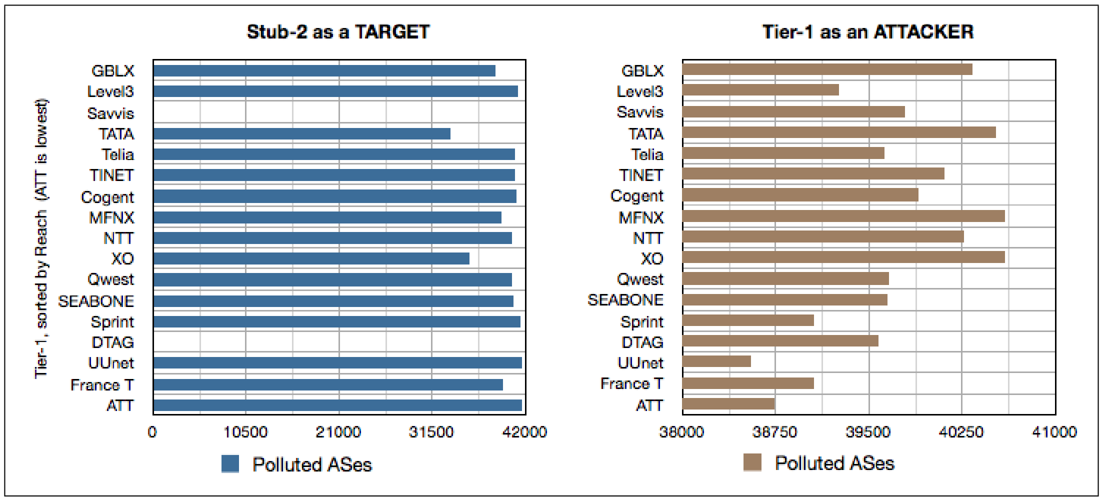


Figure 5.8. *Analysis of Tier1 attacking a Stub at depth 2 with respect to* reach

At this point we are nearing the maximum number of ASes, and very little correlation is seen between *AS reach* and the number of polluted ASes. It is now more fruitful to examine attacks from lower levels.

5.2.4. *Negative* Δ_{depth}. A stub at depth 1 attacking a Tier 1 presents an extremely simple case study. In short, all attacks fail to propagate because the attacker path length is always longer than the original valid announcement.

Similar cases at lower depths can cause varying levels of pollution due to transit ASes receiving an announcement and propagating the bogus announcement on to their customers.

5.3. *Sequential Attack Analysis*

So far we have examined variance in AS vulnerability based on statistical averages. We now show experimental results regarding this variance based on a complete set of attacks from each AS in the network. Measurements of vulnerability were performed by sequentially attacking a target AS by each of the 42,696 other ASes and recording the number of polluted ASes. Results are shown in figures 5.9 and 5.10. These graphs display the complementary cumulative sum of compromised systems for each target AS. The curve for AS 98 in Figure 5.9, for example, shows that attacks from 11,384 ASes create a minimum of 6000 polluted ASes, and that the count of attackers drops as the minimum pollution count goes up. The faster a curve goes to zero, the more resistant an AS is to attack.

The ASes in Figure 5.9 were chosen because they were all isolated within a tier-1 hierarchy. Each AS graphed is at a different *depth*, where depth is defined to be the number of hops to the nearest tier-1 AS. The depth metric was introduced in [11] and correlates with path length.

The first observation is that vulnerability increases with increasing depth, and that the concavity of the curve actually flips between depth 1 and 2. A stub attached to a transit AS attached to a tier-1 AS is much more vulnerable than a stub attached directly to a tier-1 AS.

The next observation is that vulnerability is not constant for a target AS. The attacking ASes can be strong or weak; in fact attacker aggressiveness has a strong negative correlation with attacker

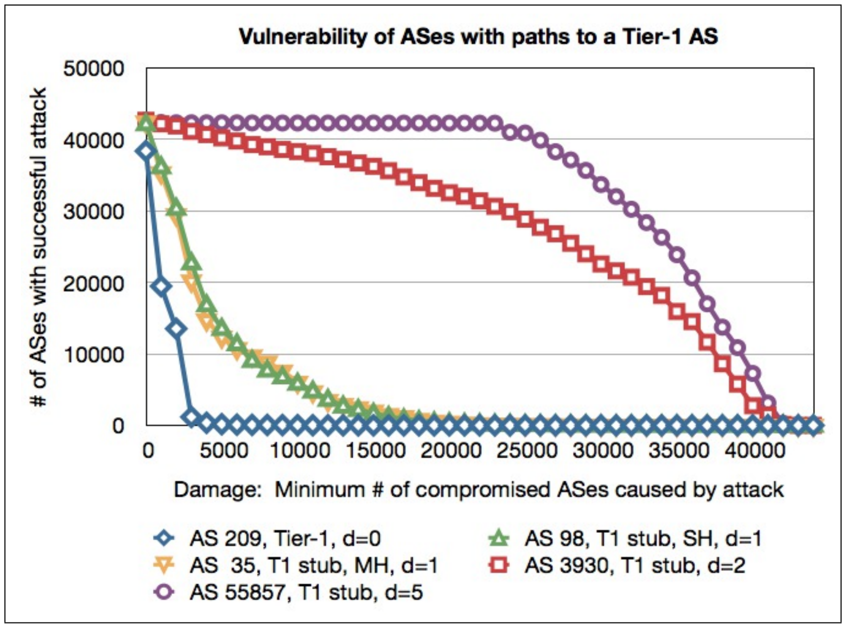


Figure 5.9. *Vulnerability analysis graph – the faster a curve approaches zero, the more resistant the AS is to attack. The Tier-1 AS curve shows high attack resistance. There is less resistance by a stub at depth 1. Multi-homing shows a very slight improvement over single-homing. The depth-2 stub shows a very large increase in vulnerability as reflected in the change in concavity, and the depth-5 AS is even more vulnerable to attack.*

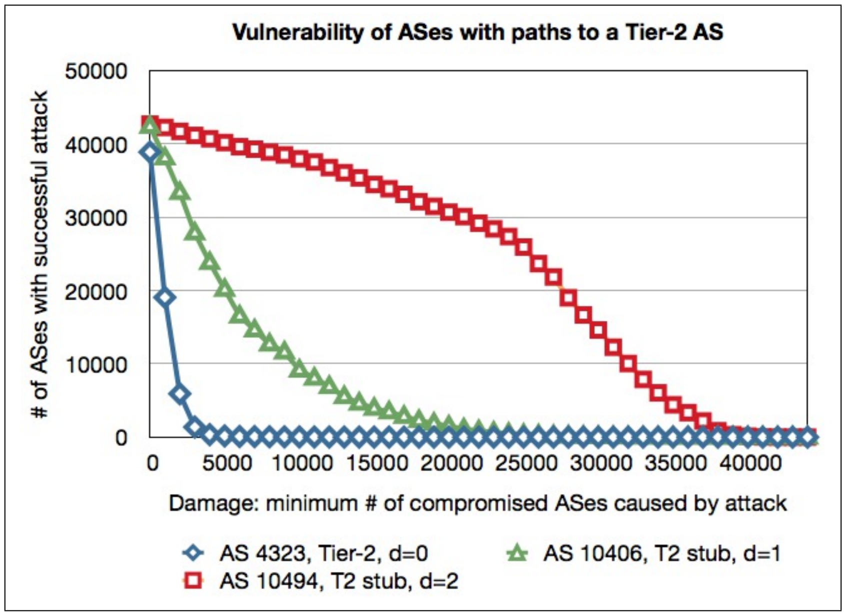


Figure 5.10. *A comparison of attack vulnerability for ASes connected to tier-2 ASes. There is similar behavior as before with respect to depth, and in fact there is a direct correspondence with the curves in Figure 5.9 when the two graphs are overlaid with each other.*

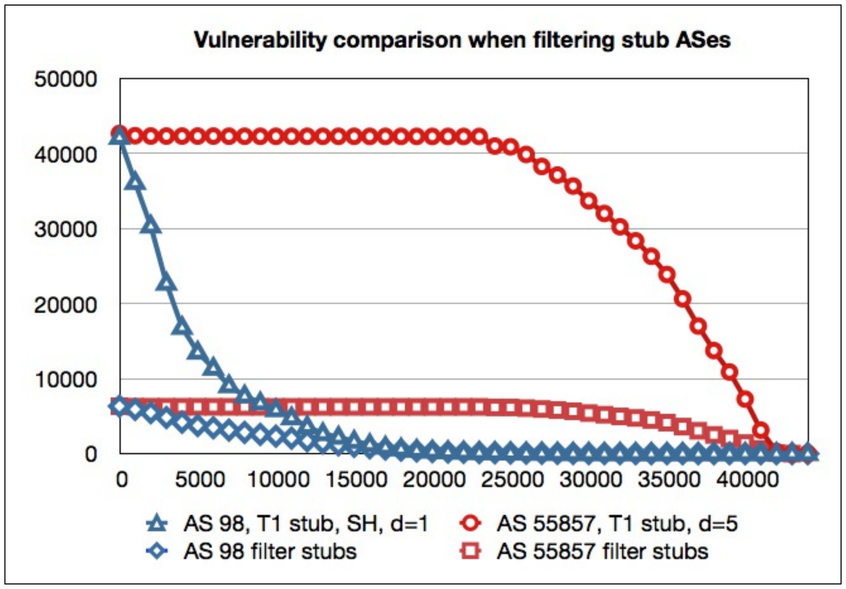


Figure 5.11. *Vulnerability of AS 98 (depth 1) and AS 55857 (depth 5) with and without filtering of attacks from stubs. Filtering simply scales the graph down; the curves retain their general shape.*

depth. Simply put, shorter paths are preferred over longer paths. A secondary factor affecting aggressiveness is the *reach* and *overlap* of the tier-1 ASes involved in the attacks where *reach* is defined to be the number of ASes that can be independently reached from AS without the aid of peer ASes [11]. There is also a very slight improvement in attack resistance when comparing ASes that are multi-homed rather than single-homed, as seen in the curves for AS 98 and AS 35.

We performed the same experiments with ASes attached to large tier-2 providers. The results are shown in Figure 5.10. These curves are very similar to the tier-1 cases, and in fact when overlaid show only minor differences. The tier-2 curve lines up with the tier-1 curve, and the remaining curves also show the same concavity as their counterparts. Normally a stub attached to a tier-2 AS would be considered to be at depth 2. However experiments show it behaving the same as a depth 1 AS. This causes us to re-define *depth* to be the number of hops from an AS to its nearest tier-1 or tier-2 provider.

Figures 5.9 and 5.10 display *worst-case* vulnerability. Another scenario could argue that transit suppliers *should* know the prefixes announced by their direct customers and defensively filter any bogus announcements from them. In reality, this doesn't always happen, but we analyze this as an optimistic case. Attacks now originate only from the 6318 transit ASes (14.7% of the total ASes). Figure 5.11 shows the curves for AS 98 (depth 1) and AS 55857 (depth 5), with and without defensive stub filters. As can be seen, the filtered curves simply scale down but keep their general shape. We shall continue using the optimistic scenario in the remainder of this paper, but remain mindful of the fact that the general situation scales back up towards worst case.

5.4. *Impact of Private Peering and Internet Exchanges*

The Internet is always growing and its topological structure continues to evolve. So far we have focused on the hierarchical aspects of the internet – Tier-1 centric and Tier-2 centric networks. However this topology is becoming inter-mixed with more complex inter domain connection schemes. The paper by Labovitz *et al* [27] points out that the internet is changing from a hierarchical set of tiers to a denser topology of direct peering between content providers and consumer ISPs. Internet Exchanges also appear more prominently in the mix. (See Figure 5.12).

The question to pose is whether this evolution make an AS more or less vulnerable to IP hijacking.

Private peering between a content provider and a set of peer ASes can actually help protect the peers from an attack on the content provider. This is due to the fact that router policy prefers a peer relationship over information provided via a provider relationship. Any attacks that arrive via the provider link will be ignored because of this preference rule. On the other hand, an attack from a customer link will pollute the AS.

A simple *before/after* experiment was conducted to prove this point.

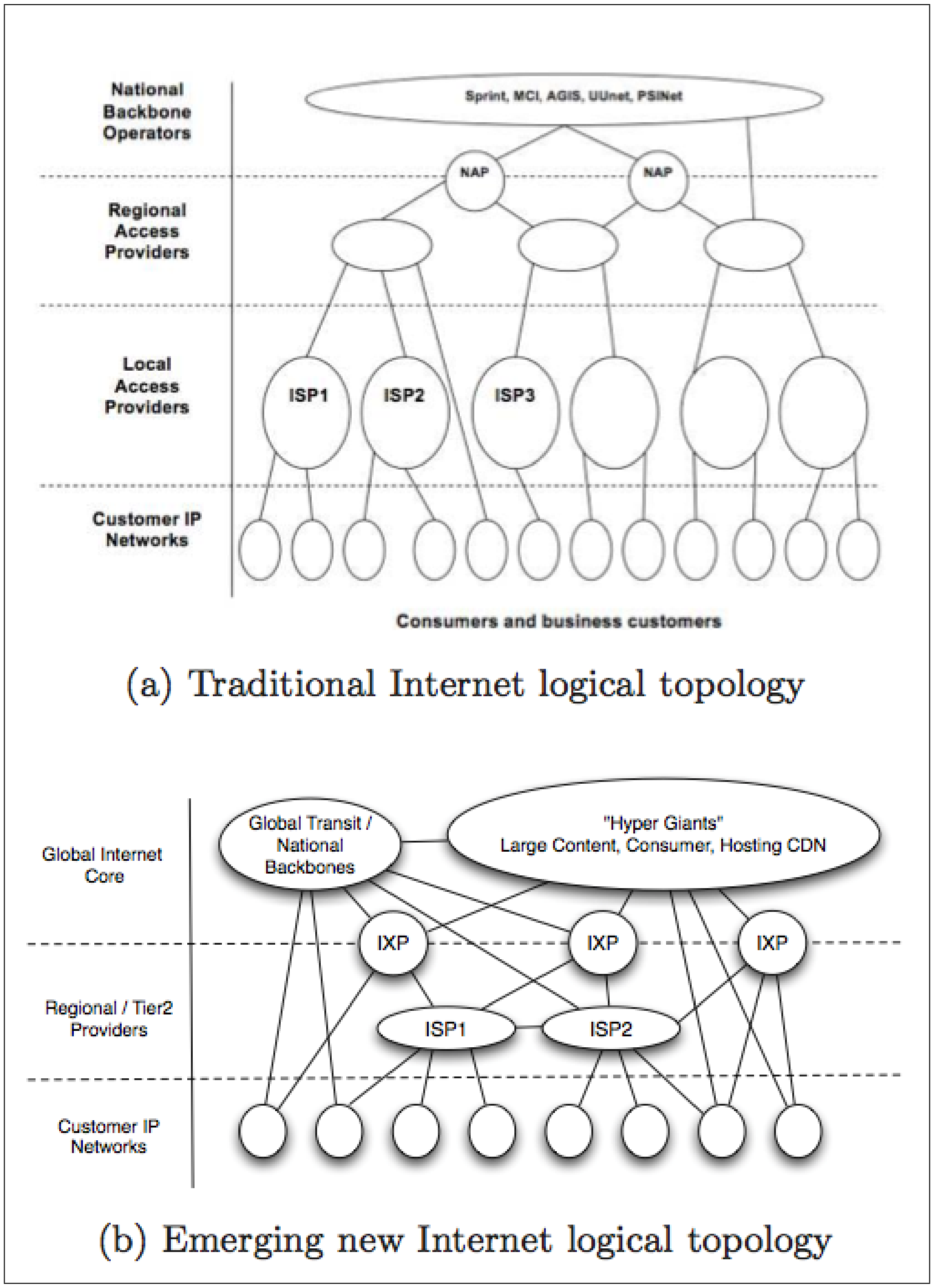

National Backbone Operators
Sprint, MCI, AGIS, UUnet, PSINet
NAP
NAP
Regional Access Providers
Local Access Providers
ISP1
ISP2
ISP3
Customer IP Networks
Consumers and business customers
(a) Traditional Internet logical topology
Global Internet Core
Global Transit / National Backbones
"Hyper Giants" Large Content, Consumer, Hosting CDN
IXP
IXP
IXP
Regional / Tier2 Providers
ISP1
ISP2
Customer IP Networks
(b) Emerging new Internet logical topology

Figure 5.12. *Evolution of Internet Topology, from [27]*

- *BEFORE:* An announcement from Google (AS 12145) was broadcast. The Colorado State University (AS 12145) received the announcement and now holds a path to Google.

We then attacked from Telia (AS1299) which is a Tier-1 provider. Colorado State University's AS became polluted and could no longer reach Google.

- *AFTER:* We added a private peering relationship between Google and Colorado State University by inserting one line into the database. When the experiment was run again, the path to Google from Colorado State remained intact.
- *BUT:* A sword cuts two ways. In a subsequent experiment we had Colorado State University attack Level3 (AS 3356) which is also a Tier-1 provider. Because of the newly installed peering relationship, Google accepted the announcement and lost its connection to Level3 in favor of CSU. This points out that NO private peering relationship should be made without a proper set of ingress and egress filters in the relationship. Typically the two parties should know the address blocks that are allowed; however if either party has a large number of customers who have their own customers, things can become very complicated. ROVER, of course, is designed to help in this situation.

We now consider the situation when there is a very high level of peering. Hurricane Electric (AS 6939) has 532 customers and peers with 2029 ASes. Attack experiments in which Hurricane became polluted caused the peers to become polluted as well.

It is all well and good that the private relationship between Hurricane and its peers helps protect their own ASes from an external attacker, but the two-edged sword can certainly create havoc if either of the parties propagates an attack against someone else. Again, proper filtering or ROVER protections should be put into place.

Internet Exchanges are simply scaled up peering relationships. Private peering is one-to-one. Internet exchanges are many-to-many. Internet Exchanges create peering islands within a topology. Since all members of the interchange are peers with one another, we can decompose the analysis to the one-to-one case and derive the same results. Within the exchange, members are protected from external attack on their ASes. But if a member should accidentally propagate a bad route to someone outside the exchange, or if a member were to initiate an attack, the high preference given to a peering relationship causes the attack to propagate to all members.

The conclusion to all this is that peering is good for the peers, but could be bad for anyone not part of the peering relationship. Filtering is essential.

CHAPTER 6

INCREMENTAL DEFENSE DEPLOYMENT

Our thesis statement is *"A system based on the existing DNS infrastructure can be deployed by a small number of institutions in an incremental fashion and still effectively thwart origin and sub-prefix IP hijacking despite non-participation by the majority of Autonomous System owners."*

We now examine this statement in more detail. In particular, we analyze the effectiveness of various incremental deployment strategies regardless of whether the technology is ROVER-based, RPKI, or even basic router filtering. We will then re-focus on the effectiveness of a ROVER-based solution in Chapter 7.

The three sections of this chapter discuss attack prevention, attack detection, and "pragmatic self-interest actions". The first sections present some unexpected results which contradict expectations regarding the effectiveness of blocking attacks by Tier-1 ISPs only. The latter section provides recommendations for deploying a solution with minimal participation by external parties.

6.1. *Deployment Strategies for Hijack Prevention*

Attack prevention implies that something exists to prevent a router from accepting and propagating a bogus announcement. Examples include prefix filters, PGBGP's use of historical data, and router firmware or other device that compares announcements to a list of authoritative route origins obtained from a secure repository such as RPKI and ROVER. RPKI uses a set of repositories for storing authorized route origins and protects them with a PKI-based certificate chain. ROVER is a method for publishing route origins in the reverse-DNS and cryptographically protecting the information with DNSSEC.

Given a mechanism for checking BGP origin security and rejecting bogus routes, how many ASes must implement this mechanism to achieve a high probability of stopping or at least minimizing an attack? Can the ASes be chosen at random or must they be methodically chosen? To address this question we ran a series of experiments against two of the typical stub ASes: AS 98, which is at depth 1 and relatively attack resistant, and AS 55857, which is at depth 5 and is very vulnerable. The experiments varied the location and number of ASes executing a mechanism for bogus route blocking. The results are graphically displayed in figures 6.1 and 6.2.

We make the following observations regarding various incremental deployment strategies for BGP security.

- **Random Deployment**: This experiment was run to simulate the real-world situation where various random ASes are motivated to deploy BGP security on their own. Over time, more and more ASes will be added to the mix. Although acting independently is commendable, this is not a very good strategy in general. As can be seen from the two charts, deploying bogus route blocking with 100 (1.6%) or even 500 (8%) of the transit ASes barely moves away from the baseline case where there are no protections. It is much more effective and certainly less costly to simply deploy to the 17 tier-1 ASes.
- **filter 17 tier-1 ASes**: This scenario was run under the assumption that the tier-1 ASes can act on their own, to everyone's benefit. As it turns out, filtering at the tier-1 ASes does improve the situation somewhat, but not enough. The average number of polluted ASes for a successful attack on AS 98 is 5084 (12% of the total ASes). Only 162 attackers are capable of pollute more than 15,000 ASes. The situation is worse for very vulnerable AS 55857. The average number of polluted ASes for a successful attack is 22,018

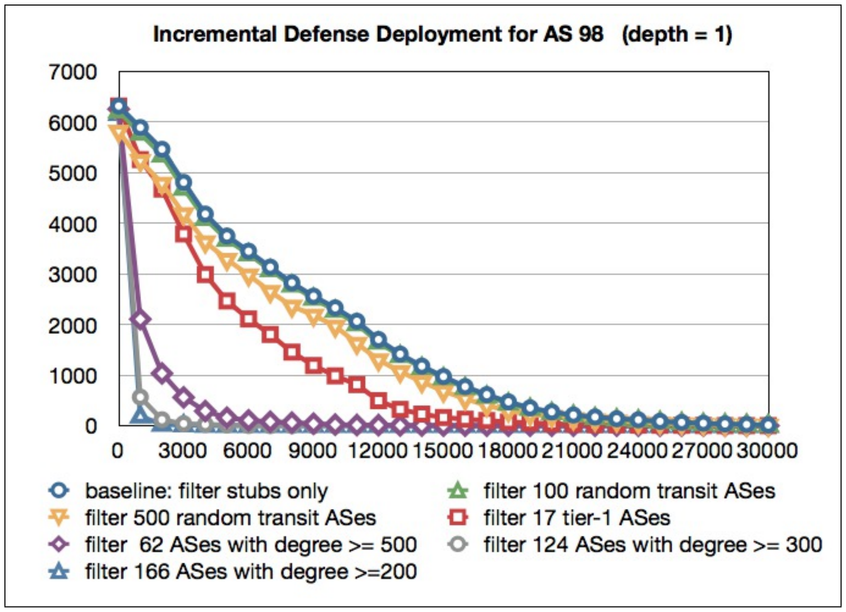


Figure 6.1. *Comparison of defensive filtering for AS98 (depth=1, relatively attack resistant). Random deployment of 100 or 500 filters has negligible to minor effect. Filtering at Tier-1 Ases gives the first real gain, however filtering the core 62 ASes with degree $\geq$ 500 shows the most marked improvement. Continuing gains are made by adding more filters.*

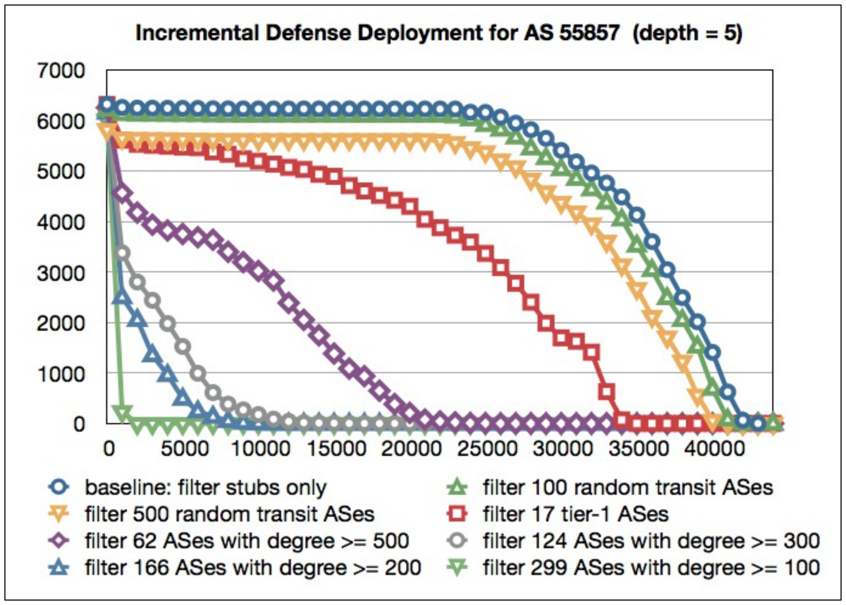


Figure 6.2. *Comparison of defensive filtering for AS 55857 (depth=5, very vulnerable). The same effects as in Figure 6.1 are seen, but starting from a much more vulnerable starting point. In this case the 62 core filters show a great improvement, and begin changing the concavity of the curve. However it still leaves the target quite vulnerable. In this case we have to increase to 299 filters before the curve shows major effect.*

(52%). Only 70 attackers pollute more than 34,000 ASes. The next strategy is to add BGP security to successively more tier-2 ASes.

- **filter 62 ASes with degree $\geq$ 500**: Filtering just 62 core ASes does give a significant gain. The average number of polluted ASes for a successful attack on AS98 is 1076 (2.5% of the total ASes which agrees with the results from [25]). However 60 attackers can still pollute more than 8000 ASes and, as expected, AS55857 is still quite vulnerable. The average attack pollutes 8562 ASes (20%) but only 42 attackers can pollute more than 22,000 ASes. As can be seen in Figure 6.2, the concavity of this curve has just flipped to form a better defensive rate.
- **filter 124 ASes with degree $\geq$ 300**: Better. Attacks on AS98 have an average pollution count of 378 and only 24 attackers can pollute more than 4000 ASes. Attacks on AS55857 have an average pollution count of 2716 and only 41 attackers can pollute more than 12,000 ASes.
- **filter 166 ASes with degree $\geq$ 200**: Even better. Attacks on AS98 have an average pollution count of 228 and only 13 attackers can pollute more than 3000 ASes. Attacks on AS55857 have an average pollution count of 1576 and only 53 attackers can pollute more than 8,000 ASes.
- **filter 299 ASes with degree $\geq$ 100**: Excellent. Attacks on AS98 have an average pollution count of 66 and only only 36 attackers can pollute more than 500 ASes. Attacks on AS55857 have an average pollution count of 163 and only 23 attackers can pollute more than 1,500 ASes. We observe, however, that despite the much improved results for AS 55857 it is likely more cost-efficient to change this target AS to be less vulnerable by

connecting to a lower-depth transit AS than it is to add security to an additional, possibly reluctant, 133 transit ASes.

Although the situation has been drastically improved it is still not perfect. A clever attacker may be limited on where attacks can be initiated and which ASes will be polluted, but can still cause some damage. Which attacks are capable of slipping by these defenses? Consider the case with 299 attack blockers deployed. For AS98, the top 5 still-potent attacks are:

ASN	Name	Pollution	Degree	Depth
11537	Abilene-Internet2	1025	87	1
25560	RHTEC backbone	895	94	1
20965	GEANT	839	61	1
20080	AMPATH Florida U	763	61	1
27750	Cooperacin	761	22	2

and for the vulnerable AS55857, the top 5 still-potent attacks are:

ASN	Name	Pollution	Degree	Depth
237	Merit	1822	37	1
46595	PVTN	1819	4	2
18592	U Desarrollo	1785	21	1
40498	NM LambdaRail	1779	13	1
3912	NMSU	1760	6	1

Thus, an attacker armed with the same tools and knowledge of which ASes have deployed BGP security can, within the limits of the simulation, plot the viability and value of a specific attack.

These attack simulations have enabled us to quantitatively analyze gains in security and also show the remaining vulnerabilities. However, the question remains: will 300 or 200 or even 100 strategic ISPs implement BGP security? This could be like *"Waiting for Godot"*, who never shows up (apologies to playwright Samuel Beckett). Given this possibility, we must consider another strategy, attack detection, which is discussed in the next section.

6.2. *Deployment Strategies for Hijack Detection*

IP Hijack Detection implies the existence of a mechanism which monitors BGP messages from a variety of world-wide vantage points and compares BGP prefix announcements to known good data. Bogus announcements caught by a detector will cause an alert to be issued.

IP hijack detectors are only as good as the quantity, topological diversity, and geographical dispersion of the vantage points (probes) they have available. Detectors are also limited by the accuracy of their comparison data. For example, detectors that use historical data can issue false alerts due to changing AS connectivity. Once again, it is prudent for ASes to securely publish their route origins so that detectors can have an accurate source of data.

Detector effectiveness can vary dramatically; some detectors with many diverse probes will capture most attacks. But we will see other detector configurations that can fail to detect large numbers of incidents, even missing some attacks that capture 25% or more of the internet.

IP hijack detectors work by collecting real-time BGP data sources by peering with routers in multiple ASes and/or by streaming data feeds from BGP data collectors such as CSU's BGPmon service [48]. The data streams are examined to compare their announced BGP prefix origins with data from a repository of trusted authoritative origins (e.g. ROVER using DNSSEC) [12] or with historical data augmented with data plane verification (e.g. ARGUS [43]). If a mis-match occurs, an origin or sub-prefix hijack is detected and operators are alerted to take corrective actions. Any particular attack may be "seen" (i.e. received and propagated onwards) by one, multiple, or possibly none of the BGP data sources which act as probes.

To analyze the effectiveness of hijack detectors, we conducted experiments on three different detector configurations. The first configuration has data probes consisting of all 17 tier-1 ASes. This was done under the assumption that a tier-1's position in the internet topology would give them wide visibility across the mesh of ASes. The second configuration was a real-world example: we used the 24 ASes monitored by CSU's BGPmon which is in turn used by several hijack detectors. The final configuration we studied has probes consisting of all 62 AS routers with degree ≥ 500. These large backbone networks are highly inter-connected and should also give good visibility to attempted hijacks.

Each of these configurations was subject to 8000 random simulated IP hijacks. Attackers and targets were chosen from the 6318 transit ASes. The results are graphed in Figure 6.3. The bar charts indicate how many attacks were detected by zero, one, or multiple probes. For example, the graph for Case 1 shows 425 of the 8000 attacks were seen by all 17 of the configured detectors. The line chart shows the average attack size vs number of probes seeing the event. This slope of the line confirms intuition; the larger the attack extent, the more collectors triggered.

The non-intuitive results we found were more intriguing, however. We make the following are case-by-case observations:

- **Case 1: 17 Tier 1 Probes**

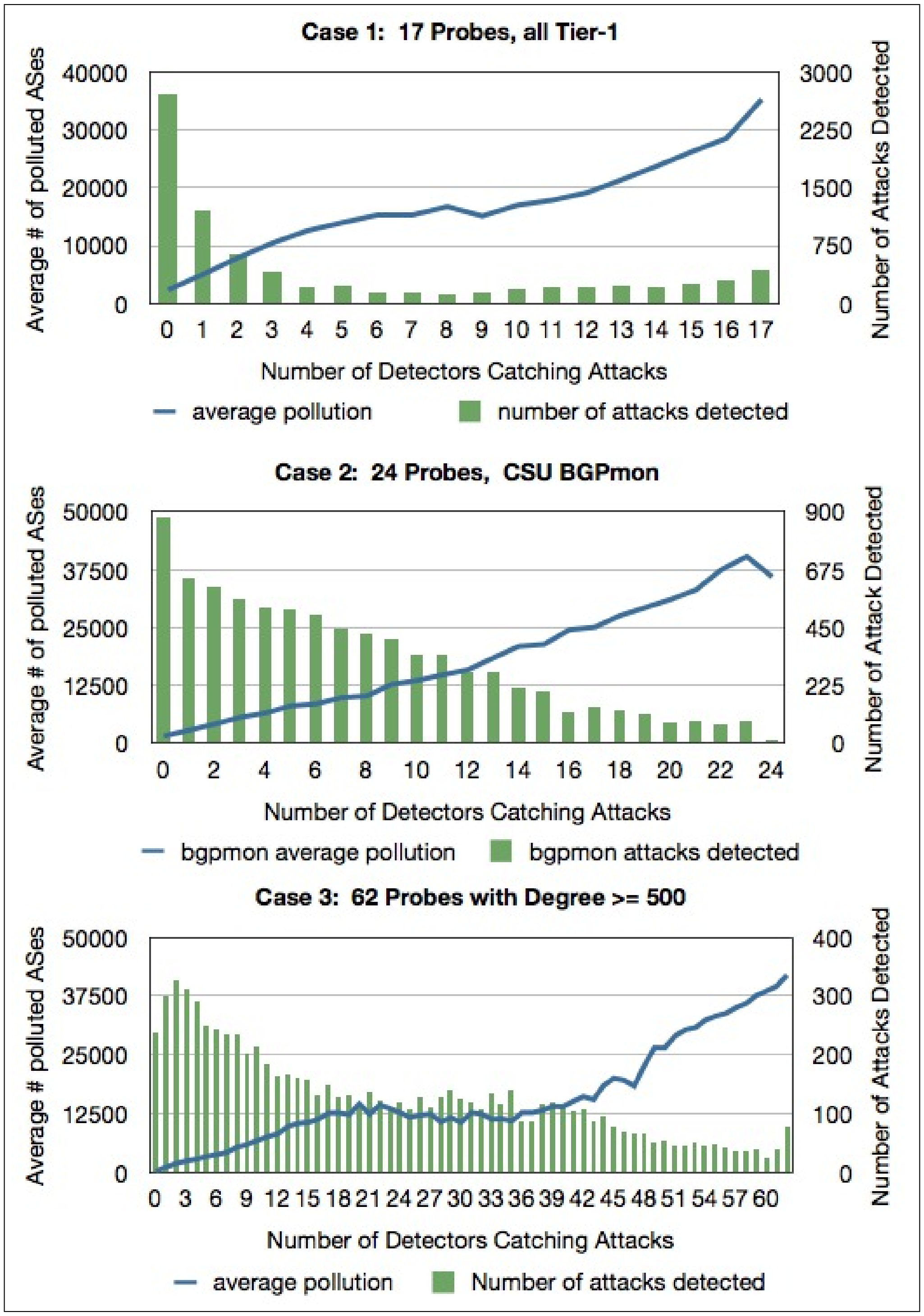


Figure 6.3. *Comparison of three detector configurations, each using different source probes. Each configuration is subject to 8000 random attacks. The "0" bar indicates the number of attacks completely escaping detection (0 probes were triggered). Otherwise 1 or more probes detected the attack. In general, the more probes triggered, the larger the attack, as indicated by the line graph. In the three examples shown, Case 1 surprisingly fails to detect 34% of the attacks, Case 2 misses 11%, and Case 3 is the best, missing only 3% of the attacks.*

Surprisingly, this configuration gives very weak detection. 2,717 (34%) of the the 8,000 random attacks completely escape detection. Undetected attacks had an average AS pollution count of 2,344 and a maximum of 20,306, (almost 50% of the internet). These are very large attacks to have escaped detection. The top 5 undetected attacks are:

Attacker	Target	Pollution Count
6450	7314	20306
21287	17228	18115
8492	50993	17879
42429	14307	17130
5715	7066	16908

- **Case 2: 24 CSU BGPmon Probes**

  This configuration is better. However, 879 (11%) of the the 8,000 random attacks still completely escape detection. The average pollution count for these “invisible” attacks is 1,521 with the maximum at 12,542. This is a surprisingly large attack size (almost 25% of the internet) to have escaped detection. The top 5 undetected attacks are:

Attacker	Target	Pollution Count
16253	4758	12542
33902	9614	11891
19806	335336	11724
197017	24228	11106
25531	132045	10769

- **Case 3: 62 ASes with degree $\geq$ 500 Probes**

  This is a much more effective configuration, but still not perfect. Only 239 (3%) of the the 8,000 random attacks completely escape detection. The average pollution count for attacks escaping detection is 202 and the maximum is 2804 (attack extent at 6% of the internet). The top 5 undetected attacks are:

Attacker	Target	Pollution Count
11014	34243	2804
28135	26210	2478
17696	12895	2286
2153	198704	1886
47602	56247	1792

How is it that AS 6450 can hijack AS 7314's address space, pollute over 20,000 autonomous systems, and still not be detected by the 17 tier-1 probes configured in Case 1? AS7314 is at depth 1 and is multi-homed to tier-1 provider 7018 and a medium sized depth-2 provider, AS 12083. This results in all of the tier-1 ASes having a path length of 2 to AS 7314 (with the exception of AS 7018 having a path length of 1). The simulator's policy for tier-1 ASes is to prefer shorter path regardless of whether the connection is from a peer or customer. Attacker AS 6450 is at depth 2 so it can't replace the existing paths at the tier-1 ASes. Furthermore, AS 6450's providers are AS 6939, AS 4436 and AS 22822, all which peer to thousands or hundreds of other ASes. This causes high attack propagation. If tier-1 policy were different, then some of them may have detected the attack.

The smaller undetected attacks observed in configuration three are not straightforward to analyze. Animations of the polar graphs showing the attack propagation give some insight, but basically all we can say at this point is that the small amount of traffic during the attack propagation never hits one of the probes. This is an area of future research.

Again, the numbers from the simulation should not be taken verbatim, but do provide general guidance and insight. From this set of experiments it is logical to recommend that BGP detectors peer with as many high-degree, non-overlapping ASes as possible, rather than with random ASes, to maximize their effectiveness. The currently available coverage with BGPmon by itself is insufficient for good detection, and we encourage more effective peering relations be established.

6.3. *Pragmatic Self-Interest Actions*

Bruce Schneier states, *"Security is a process, not a product"*. BGP security will not happen in a single step, it will take a series of actions over time. But how much time? Rather than sit and wait, responsible organizations can start to take pro-active actions immediately that improve their own security and help others move this process along.

We next propose an approach that can help an organization reduce risk with respect to IP hijacks. The process could be used by a concerned AS for increasing security for important customers or by a regional advisory board to make recommendations regarding security of critical infrastructure.

The general approach is to execute the following steps:

- analyze the relevant AS topology
- reduce vulnerability
- publish route origins and recruit other ASes to publish
- incorporate filters based on published data

- use detection

Analysis: The first step is to examine the subset of relevant AS topology, listing end-point ASes between users and services. Prioritize goals, don't try to solve everything at once. As an example, a bank may have world-wide customers whose internet traffic traverses multiple ASes, but the bulk of the traffic is isolated to one geographic region. Start with that region and map the ASes involved. Measure *depth* to assess potential vulnerability.

Reduce Vulnerability: The depth analysis may reveal some ASes to be more vulnerable than others. If possible, increase resistance to attack by re-homing and multi-homing these ASes to reduce *depth* , and to increase non-overlapping *reach*. If the AS is out of immediate direct control, at least inform the AS of their potential vulnerability.

Publish route origins: This is a critical step. Publishing authoritative route origins in ROVER and other secure repositories will enable accurate real-time detection as well as provide good data for building prefix filters. Eventually this published data will be used in new routers and devices to block bogus routes in real-time. But in the meantime, The simple act of publishing creates leverage: as more organizations use detectors and filters based on published data, the more accurate and comprehensive they will become.

Filter: Build prefix filters. Filtering is one of the only practical blocking technologies available today. Filters should be based on authoritative data. Block the known prefixes of immediate customers and set egress filters for your own address space. Add filters from securely published route origin sources; initially these can be chosen based on the AS topology analysis. The list could grow to include all published data At some point, however, this process may no longer scale and this will encourage the incremental adoption of other hijack blocking technologies.

Table 6.1. *Local Protective Actions for AS55857*

Action	# of Polluted ASes from Internal NZ attacks (186 attackers total)	# of Polluted ASes from External attacks (200 random attackers)
none	113 (60%)	28 (15%)
Filter at VOCUS	74 (40%)	26 (14%)
Re-home up 2 levels	46 (25%)	12 (6%)

Use detection: Subscribe to a service or run a local IP hijack detection program. But be an active participant. Understand the set of probes used in the detector and run simulations to see if there are any blind spots regarding relevant AS endpoints. If necessary, determine new probes that can improve detection accuracy. Have an operational plan if an alert is generated.

Several experiments were conducted to validate these recommendations. We again used AS 55857 as an attack target because of its vulnerability. This AS is located in New Zealand, along with 186 other ASes. We wanted to see if IP hijacking could be reduced just within the NZ region. The first experiment re-homed AS55857 up two levels. In the second experiment we added a single prefix filter to VOCUS, at AS 4826, based on the AS topology.

In the re-homing experiment, we reduced the average number of compromised NZ ASes from 113 (60%) to 46 (25%) based on attacks generated from each of the 186 ASes within the region. We also ran a sample of 200 attacks from outside the region. In this case the average number of compromised NZ ASes dropped from 28 (15%) to 12 (6%). In the filtering experiment we reduced the number of compromised ASes caused by regional NZ attacks to 74 (40%). Attacks from outside the region resulted in an average of 26 (14%) compromised NZ ASes. These data are summarized in table 6.1.

Many more experiments could be conducted to determine if further improvements are possible. This is an area for future investigation.

CHAPTER 7

ROVER SYSTEM DESIGN CONSIDERATIONS

In this chapter we examine several important design issues that can affect ROVER-based applications. We also discuss criticisms and objections to ROVER put forth by RPKI adherents.

The most important design consideration focuses on system reliability and resilience. How reliable can a ROVER system be, given its dependence on a potentially unreliable DNS service? Furthermore, how do we know if the DNS is even capable of handling the additional load posed by many ROVER devices? Will the system scale?

An analysis of these topics raise related questions. In addition to the DNS, what are ROVER's other external dependencies? What is its threat model? Can a ROVER system be made fault tolerant?

To instill confidence in a complete ROVER-based solution, we must go beyond basic mechanisms and examine the surrounding eco-system. We must analyze potential failures and threats, as well as day-to-day operational issues. A ROVER solution must be *robust*; that is, designed to prevent failure. It must also be *resilient*, accepting that failure is likely to happen and is therefore designed to detect failures and recover quickly from them. The following topics are examined:

- Threat Model and Failure Modes
- Operational Considerations
- Global DNS Capacity and Load
- ROVER criticisms and objections

The analysis and research experiments conducted in these areas have resulted in improvements to the fault-tolerance of ROVER. More importantly, we demonstrate that the underlying DNS infrastructure is fully capable as a method to publish and retrieve route origin data.

7.1. *Threat Model and Failure Modes*

This section presents an analysis of the vulnerabilities and potential failures related to a ROVER system's external dependencies, specifically communication with BGP, DNS and the remote subscriber. We will ignore hardware and operating system vulnerabilities as these types of issues are well-documented elsewhere. We assume that the device has been built on a secure (or at least hardened) operating system. Only the necessary ports have been opened, the system securely boots, etc. The areas examined are:

(1) vulnerabilities with respect to BGP communication

(2) vulnerabilities with respect to DNS communication

(3) vulnerabilities with respect to subscriber communication

(4) system redundancy and fail-over

These four vulnerabilities are illustrated in Figure 7.1 with respect to an IP Hijack Detector using ROVER technology. The small "devil" symbols indicate either an attack, such as a BGP man-in-the-middle spoof or a DDOS attack, or a component failure, such as a DNS server being offline and not returning a response to a DNS query.

Attackers craft exploits to take advantage of vulnerabilities. Example exploits include:

- If BGP communication can be blocked, a ROVER hijack detector may not see a bogus announcement and no alarm will be issued.
- If an attacker can spoof the DNS traffic between ROVER and its resolver, all sorts of problems can occur. This last communication segment is *not* protected by DNSSEC, so false messages would not be detected. An attacker could trick ROVER into thinking an attack happened when there was no attack, or conversely, mask a real attack by responding with a fake SRO verifying the fraudulent origin. This invalidates all system reliability!

- If a ROVER hijack detector can't send an alarm to a subscriber because an attacker has DoS'ed the communications link, the subscriber obviously won't get the alarm. ROVER may have done all the correct hijack detection, but fails to alert the affected parties.

The remainder of this section examines each of the four vulnerabilities using an example of a ROVER-based IP hijack detector. The discussion will describe the basic threat or failure mode, suggest possible defenses, and describe the results of any research experiments performed that validate the defenses.

As always, there may be additional vulnerabilities and attack vectors that are unanticipated by the designer. Security audits, penetration testing, and successful attacks will identify subsequent protections and security measures that should be undertaken.

7.1.1. *Vulnerabilities with respect to BGP communication.* **Vulnerabilities:** A ROVER hijack detector is completely dependent on receiving a constant real-time stream of BGP announcements from each of its remote vantage points. These announcements may come from multiple routers that have set up a peering arrangement with ROVER. Additionally, BGP announcements may also come from a BGP message aggregator such as BGPmon [48] which streams data to users via an XML telnet session.

- *BGP Peering Issues:* The Huston survey article [24] outlines multiple BGP session vulnerabilities. There can be attacks with respect to authentication, spoofing, message alteration, TCP replay, man-in-the-middle attacks, etc. Denial of Service is another attack vector as BGP is built on top of the TCP protocol, and could be subject to SYN floods and data floods.
- *BGPmon Issues:* BGPmon is a very practical source of BGP announcements. It provides a single real-time stream of XML encoded BGP data from multiple world-wide BGP

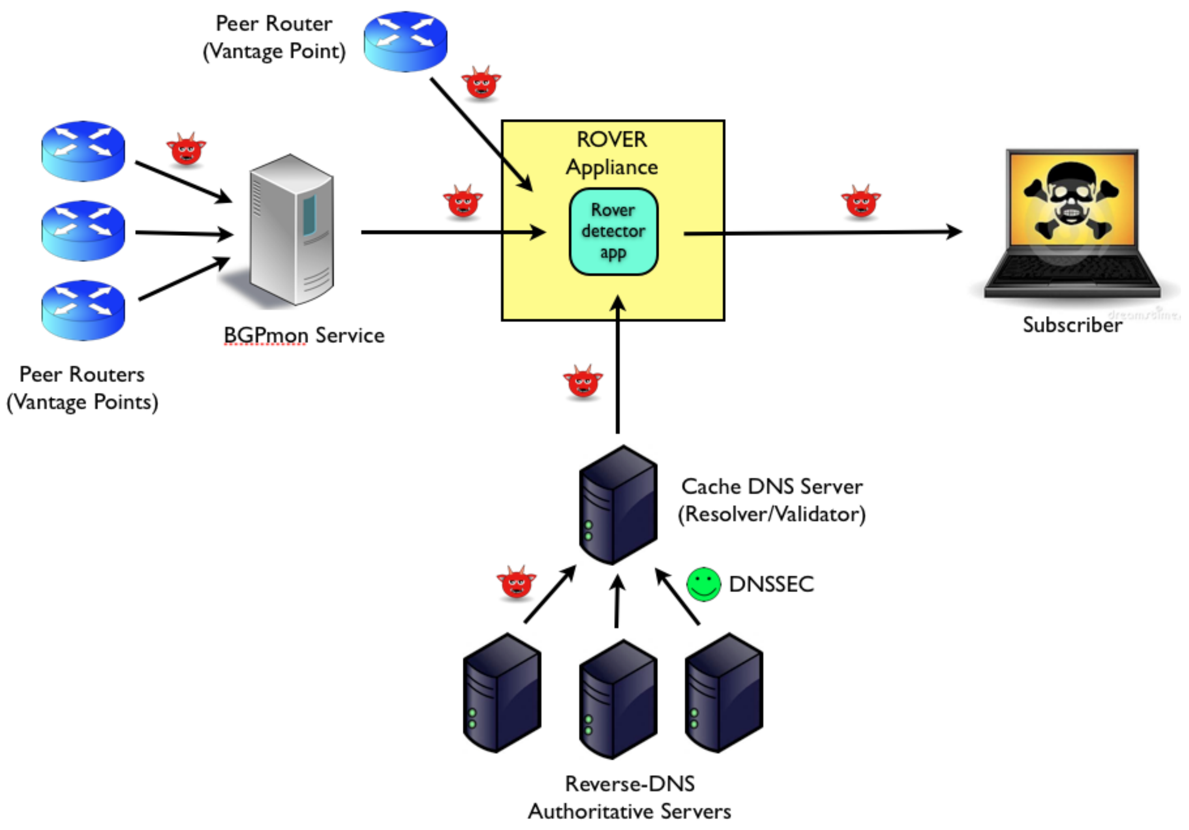


Figure 7.1. *Block Diagram of ROVER-based IP Hijack Detector showing Potential Vulnerabilities*

router peers. One of its strengths is its ability to chain multiple services together into one consolidated real-time data stream that removes redundant messages. Unfortunately, there is no strong authentication associated with BGPmon. The connection between the XML data stream and ROVER is via the *telnet* protocol. *Telnet* is not secure and can be easily spoofed or altered with a man-in-the-middle attack. A stronger authentication scheme is a necessity for a robust hijack detector.

Solutions: To protect against these attacks, the basic BGP session defenses described in [24] should be incorporated into the ROVER detector device. A minimum level of defense is to use TCP

MD5 shared-key authentication for all BGP peering. GSTM (generalized TTL Security Mechanism) would add an additional level of security.

Some operating systems have network stacks with built in DDoS defenses. Otherwise the ROVER application should log and alert for any disruption to BGP data transfers, keep-alives and session flapping.

The BGPmon communication method could be improved via data authentication techniques. However, an alternative defense against these attacks is to simply run a separate copy of BGPmon directly inside the same system that the ROVER hijack detector is running. This internal copy of BGPmon could maintain its own set of router peers, and could also chain to other BGPmon collectors, identifying itself with its IP address and authorized via access control lists and other authentication mechanisms. Furthermore, this internal copy of BGPmon now provides its XML data stream to ROVER via telnet over `localhost`, avoiding external network communication.

A separate reason to incorporate a BGPmon application within the ROVER device is to enable redundancy and failover. A ROVER hijack detector *must* have a continuous and reliable real-time source of BGP data. If it is possible for the the data stream to be stopped or altered, the ROVER hijack detector will be considered to be unreliable.

The most likely failure mode for an external BGPmon collector is to simply crash, or for its network connection to drop or to be subject to a denial of service attack. From time to time the BGPmon collector will be taken down for preventive maintenance. To accommodate these situations, an internal copy of BGPmon could chain to multiple geographically distributed redundant copies of the external BGPmon collectors. Any single device could fail, but the data stream from the other devices would still be available. Redundant data is removed by the BGPmon chaining operation, keeping the XML stream to ROVER as straightforward and simple as possible.

Experimental Results: The ROVER testbed was modified to incorporate a BGPmon application within the Linux server running the detector application. This proves feasibility of chaining to an external BGPmon service. No other peering or security experiments were conducted.

7.1.2. *Vulnerabilities with respect to DNS communication.* **Vulnerabilities:** Another critical dependency for a ROVER-based hijack detector is in its communication with the reverse-DNS. In general, the global DNS provides a robust and resilient publishing and data retrieval mechanism. DNS is a mission-critical component of the Internet and is designed with redundancy and resiliency in case of server or subnet failure. This is primarily handled by duplicating DNS data across geographically dispersed slave servers. Data integrity is ensured via a DNSSEC cryptographic signatures. Nevertheless, DNS doesn't always work and there are several failure modes that need to be considered:

- The DNSSEC "last mile" problem:
  - A DNS answer is properly validated by the external DNS Cache server shown in Figure 7.1, but the insecure communication link between the DNS Cache server and the ROVER device has been compromised. A forged answer will be accepted by ROVER because there are no integrity checks at this point.
- No response, or SERVFAIL response, to a DNS query
  - A DNS query fails to return an answer and *times out*. This could be due to DDoS attacks at any of the authoritative servers or at the DNS Cache server illustrated in Figure 7.1. It could also occur due to a server simply being unavailable. (Note: This is *not* the same as DNS returning a validated response saying "no data available" or NXDOMAIN. These situations still return responses, not timeouts or SERVFAIL.)

  - A DNS query is directed to a misconfigured server and returns a SERVFAIL.
  - DNSSEC validation fails because of a spoofed man-in-the-middle attack and returns a SERVFAIL.

Solutions: The "last mile" problem is easily solved by simply not using an external DNS cache, thereby eliminating it as a dependency. Instead, the DNS caching resolver and DNSSEC validator should be physically contained inside the same appliance and operating system as the ROVER application itself. This completely eliminates the "last mile" because communication between ROVER and the DNS cache will be performed over `localhost` rather than some external network link.

Using an internal resolver has a side benefit as well: it eliminates the possibility of a communication path failure between the ROVER application and an external resolver. UDP packets are unreliable by nature and it is quite likely that responses to a DNS query will be dropped or altered by an external network link. The internal resolver configuration is therefore more robust.

The "no response" issue can be addressed with a variety of mechanisms, each strengthening the overall system. Mechanisms include DNS caching, ROVER database, DNS Cache preloading, DNS Cache prefetching, threading for asynchronous parallel DNS queries, and exponential re-try mechanisms after a DNS query failure.

- *DNS Caching:* The first solution is to take advantage of the caching mechanism contained in the DNS resolver. Responses received by a resolver will be stored in internal cache memory until the response ages out (TTL expiration) or is forced out due to other caching activity. This suggests that the DNS cache SHOULD be large enough to hold all the DNS responses associated with a complete router RIB table.

Storing a ROVER record in the DNS cache provides two benefits. Performance will improve because data can be retrieved in microseconds rather than the typical time of 100 to 500 milliseconds or even longer when timeouts occur. Furthermore, if the data is long-lived, the data can be retrieved from cache even if the authoritative source of the data is under a DDoS attack.

TTL's are typically in the range from minutes to hours, depending on how stable the data is. The TTL for NS records at `in-addr.arpa`, for example, is 1 day. The TTL's at ARIN are typically 1 hour. The TTL for `colostate.edu`'s PTR record is 10 minutes. Note, however, that these examples are no guarantee of long life. At some point the data will no longer be in the cache and a ROVER DNS query will have to communicate with the external DNS authoritative servers. Although a response is usually received, there is no guarantee.

- *ROVER database:* To enhance reliability, all ROVER-based systems should maintain an internal database storing information for each IP prefix that it has processed. This database provides a backup mechanism in the event that a DNS query fails to retrieve an answer. It is also a mechanism to enumerate known ROVER data and enable iteration. New items are added to the database as ROVER processes BGP announcements containing 'never been seen before" IP prefixes.

  The database entries are indexed by IP prefix. The data associated with the IP prefix includes a a timestamp indicating the last time the data entry was written, a flag indicating whether an SRO record, RLOCK, or no ROVER data exists for that prefix, the SRO data fields or RLOCK prefix, and the DNS TTL. The TTL is informational only; it is not used to remove entries from the database.

The ROVER database will be empty the first time a ROVER device is started up. To "prime the pump", DNS queries can be issued using a list of IP prefixes obtained from an Oregon RouteViews RIB table [38]. These queries will load information into both the ROVER database and the DNS cache.

Although this database may seem redundant to the DNS cache, it actually contains extra information not present in the DNS cache. In addition, the database is persistent in that it never expires and data entries are refreshed as new BGP messages are processed.

The database is used in cache pre-loading and in the event of a DNS timeout or SERVFAIL, as will be described below.

- *DNS Cache Preloading and Reloading:* To increase performance, and to potentially avoid future DDoS problems, it is wise to preload the DNS Cache on system re-start. This can be done by performing DNS queries on each and every known prefix by iterating through the ROVER database. As new data is retrieved, the ROVER database is also updated.

  Cache "reloading" may also be useful as a reliability mechanism. Every N hours another iteration can be performed to issue DNS queries which refresh the DNS cache and database.

- *Prefetching:* This is a mechanism available in some DNS cache servers (e.g. *UNBOUND* and *SECURE64*) to aggressively maintain fresh data in the cache. Pre-fetching is triggered whenever a query is made for a record that is about to expire from the cache, typically within the last ten percent of its TTL. The pre-fetch mechanism issues a new set of authoritative queries to refresh the cache and reset the TTL even though the original cache entry hasn't expired yet. The requestor's query is answered immediately, of course, because it is a cache hit. The end result is that“popular” DNS records are always available in the cache.

This mechanism has limited use for ROVER. One benefit is that performance is faster for frequently queried IP prefixes. Additionally there is a remote possibility that pre-fetch can help in the event of a TIMEOUT or SERVFAIL. Consider the case where a DNS record has a 24 hour TTL. If a query is made within the last 2.4 hours (last ten percent of the TTL), the query will get the cached response and the DNS server will try to pre-fetch new data. If A DDoS attack is occurring, that request is likely to TIMEOUT. But the data is still in the cache and will be available for 2.4 hours. If the DDoS attack stops within that time window, no harm will be done. The next query will get data.

Although pre-fetch helps in this particular sequence of events, is a very unlikely scenario. Better mechanisms to deal with TIMEOUTs and SERVFAILs are needed instead.

- *Asynchronous queries:* Experiments with the ROVER testbed demonstrated a wide range in the incoming query rate of BGP announcements. At times there were only 5 announcements per second. At other times there were 500 or more incoming BGP announcements per second. Each of these incoming announcements requires an associated DNS query.

  These queries should not be performed serially. Although most queries will be answered instantaneously (if in cache), or within a few hundred milliseconds (if a cache miss), some queries will not be answered at all and the DNS server has to wait for a pre-determined timeout. The DNS server then re-queries the same authoritative server that failed as well as alternative authoritative servers. The entire process can take seconds until the DNS server finally gives up and a final timeout is returned to the ROVER application. This is far too long to wait, as hundreds of other BGP announcements will potentially queue up awaiting their turn to do a DNS query. If several more time out, the entire system may never be able to catch up.

The solution to this problem is to do the same thing that a DNS caching resolver does: perform asynchronous queries via threading or some other parallelization mechanism. One method is for the ROVER application to use hundreds of worker threads to process an announcement and perform the associated query. This keeps the system running smoothly. If any particular query gets bogged down, the other announcements are still able to be processed by the other workers.

- *Exponential Retry:* Despite all of the mitigation mechanisms described above, we finally must face up to the situation where the ROVER application simply fails to receive a reply, and instead receives a TIMEOUT or a SERVFAIL. The ROVER algorithm requires that the BGP announcement be accepted as INSECURE (neither proven BOGUS, nor proven SECURE).

  But what if the timeout were due to a DDoS attack or to a temporary failure? A query performed at a future time may prove that the BGP announcement is SECURE or BOGUS. How can this be handled?

  Two different methods are possible, exponential query re-try and using the ROVER database as a backup mechanism.

  – Exponential retry involves placing the failed announcement on a special queue which retries the query multiple times, typically with exponentially slower intervals between queries. In the case of a SERVFAIL, the timing interval may have to be long because many DNS caching servers typically store a SERVFAIL for five minutes. Asking any sooner just results in a cache hit and returns SERVFAIL immediately. If the DNS Cache server has a configuration variable for SERVFAIL TTL, it should be set to zero.

If, after multiple retries, there is still no response, an IP hijack detector will simply have to give up and accept the INSECURE announcement. On the other hand, if it receives a response indicating BOGUS, it can still issue an operator alert, although it will be somewhat delayed. The message to the operator should indicate that this is a delayed alert and contain the timestamps for both the original announcement and the time it was actually detected.

We have been primarily discussing IP hijack detectors. The situation for a ROVER application directly connected to a router to prevent it from accepting BOGUS announcements is somewhat trickier. If the router has already accepted the BOGUS announcement, and later finds out that the announcement was BOGUS, there met be some ”undo” mechanism which forces the router withdraw the announcement.

– The second mechanism for dealing with a TIMEOUT/SERVFAIL is to take advantage of information in the ROVER database described earlier. If a query fails to return data, ROVER can examine the database to determine if a prior query for the IP prefix had successfully returned an SRO or RLOCK. If not, then returning a response of INSECURE is appropriate since the previous response was INSECURE. On the other hand, if the previous query had successfully retrieved ROVER DNS records, then this is a strong indication that something is wrong.

Normally one would expect to retrieve the same data, updated data, or an indication that the data had been removed. A timeout or SRVFAIL could now be interpreted as an attack.

In this situation ROVER should examine the timestamp of the record and its TTL. If the record is still within its TTL limit, it should use this backup data to validate the record just as if it had come from the DNS. If the record has expired, ROVER could issue an alert to its subscribers indicating "high probability" for an attack (as opposed to its normal message indicating "definite hijack attempt"). Operators could then take manual steps to determine the root cause of the incident. It is likely the subscriber is having other problems if their DNS is being attacked.

Finally, if there is no data in the database, then this is a newly announced prefix. A warning should also be issued in this case.

Experimental Results: The ROVER testbed was modified to incorporate an internal DNS resolver to eliminate the last mile problem as well as UDP communication failures. Experiments with a lossy network to an external cache resolver resulted in many UDP drops causing query timeouts. These were eliminated when the resolver was incorporated within the ROVER appliance.

Exponential retry was also implemented. The data networks remained lossy for seconds at a time, indicating that the retry mechanism should be maintained for several minutes.

Experiments were performed to iterate through a RouteViews RIB to perform 400,000 queries to preload the DNS cache. This took approximately five minutes using 200 parallel threaded queries. This could be easily shortened by provisioning more worker threads. Further details of this experiment are provided in Section 7.3.

The ROVER database was not directly implemented because there were too few actual ROVER records in the live Internet. As a related experiment, however, a comparison to a historical database was implemented. ROVER issued different messages depending on whether the data failed the actual ROVER test, or whether it failed a historical comparison. This experiment led to the evolution of the ROVER database idea.

7.1.3. *Vulnerabilities with respect to Subscriber Communication.* **Vulnerabilities:** The communication link between an IP hijack detector and a subscriber can be compromised in two ways: fraud and outage. Fraud can occur via a man-in-the-middle attack in which messages between the detector and the subscriber are altered. An outage can occur due to DDoS, network failure, or by an actual IP hijack of the address space for the detector or the subscriber – the detector is taken out by the very method it is supposed to detect!

Solutions: Both of these issues can be solved by defining a secure communication protocol between the detector and the subscriber. The ROVER testbed, for example, sends data to its subscribers via a `telnet` connection which issues an XML data stream. `Telnet`, however, does not perform any authentication and has no provision for data integrity. This can be solved by either replacing `telnet` with a secure connection protocol (`SSH`) or by encrypting the XML data stream.

`SSH` would require login credentials at the ROVER appliance for each subscriber. Doing this, however, creates more problems than it solves; for example, a user with login credentials could exploit OS vulnerabilities and wreak havoc. Therefore we propose a method using data encryption to solve authentication and data integrity:

- ROVER publishes a public key in the DNS using a DANE resource record as specified in IETF RFC 6698 [20] . This makes the public key accessible to anyone who wants to subscribe.
- When a new connection is established, ROVER generates a symmetric key, encrypts it with its private ROVER key, and sends this encrypted key as the first XML message to the subscriber. This is similar to the method used by web browsers communicating via SSL.
- All subsequent XML messages are encrypted with the symmetric key.
- The subscriber uses the public key to decrypt the symmetric key. If everything is valid, the remaining messages will decrypt properly. If a fraudulent or altered message arrives, it will not decrypt and an alert can be logged and sent to the subscriber.

The key (pun intended) to solving authentication and data integrity basically lies in ROVER maintaining a well-protected private key and associated public key to establish its unique identity.

We now examine DDoS, hijacks, and other issues which stop messages from arriving at a subscriber. Earlier papers, notably PHAS [28], attempted to solve this problem by sending alerts to a subscriber via multiple email addresses. This technique assumes that at least one of the subscriber's mail servers, perhaps one in a separate address block, would still be accessible. Other suggestions have been to send notifications via technologically differing mechanisms such as email and SMS text messages.

None of these solutions is particularly robust. As an alternative, we propose a mechanism based on communication using the XML data stream. Each subscriber must run a local "XML Receiver" program to parse XML messages and take appropriate actions:

- After parsing a message, the receiver discards the message if it is irrelevant to the subscriber (e.g. a hijack of someone else's address space may not be of interest).
- The ROVER detector sends periodic "keep-alive" messages as part of its XML stream.
- The receiver sets a session timer whenever it receives a "keep-alive". If this timer should expire before receiving another "keep-alive", a communication failure has occurred. This failure event should be logged and an alert sent to the subscriber.
- Messages of interest, such as a hijack of an address owned by the subscriber, can then send an alert to the operator by any desired technique, including SNMP traps, SMS messages, email, etc.

In other words, this "keep-alive" technique uses the absence of a message to detect a broken communication path. If the communication path becomes compromised, the subscriber will be alerted and can perform checks to see why the link is broken. This may be an IP hijack, DDoS attack or other problem with the network. In any event, action can be taken.

Experimental Results: The ROVER testbed was modified to issue periodic "keep-alive" messages and the receiver was able to parse and process these messages. No experiments were performed with encryption, as these are well-understood capabilities.

7.1.4. *Redundancy and Failover.* **Vulnerabilities** A *single point of failure* is any part of system that stops that system from working if the component should fail. A fault-tolerant system must not have any single points of failure. Furthermore, a robust system should accommodate component outages due to preventive maintenance, upgrades, etc.

An examination of the block diagram in Figure 7.1 identifies several points of failure. BGPmon could go down, ROVER could be turned off for maintenance, and so forth.

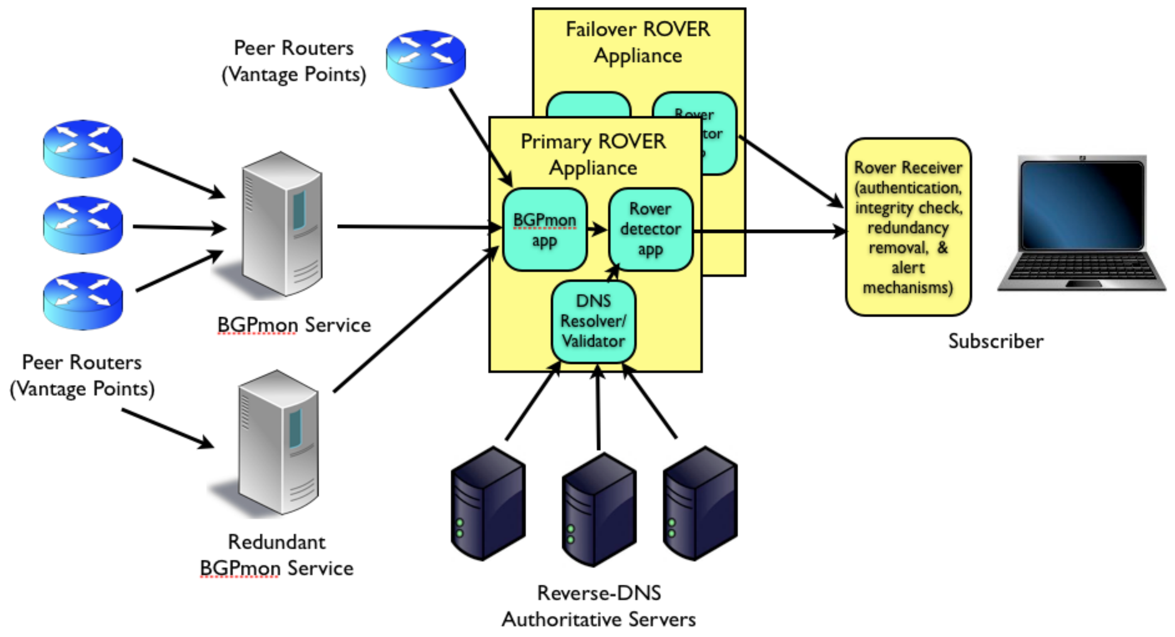


Figure 7.2. *Block Diagram showing improvements for security and redundancy*

Solutions An improved block diagram with redundant components is shown in Figure 7.2. In this diagram there are two BGPmon services, and a failover ROVER appliance as well. These ROVER devices may act in parallel, rather than failover mode, since the subscriber receivers are capable of removing and de-duplicating messages from its senders. Although not shown in this diagram, it may be useful for the subscriber to have redundant XML receiver programs.

Diagram 7.2 also shows improvements to the ROVER device that have been discussed earlier. Each ROVER device has an internal DNS cache resolver/validator, as well as an internal BGPmon program that can directly peer to external router vantage points and chain to BGPmon services.

Experimental Results No experiments were conducted for redundancy. This topic is well understood in general and is included here for completeness.

7.2. *Operational Considerations*

Several design improvements to the ROVER resource records were implemented after analyzing an operator's work flow for inserting, removing and transferring route origin information.

When an operator first publishes route origins in the reverse-DNS, it is likely that mistakes are going to be made. If so, there is a risk that ROVER devices will access the erroneous data and perhaps block routes or cause false alarms at detectors. To prevent this, the operator needs to be able to try experimental publishing that poses no risk.

Another issue involves changing a prefix from one AS to another. An operator cannot simply change the published SRO statement and update routers. The old SRO statement may still be cached in ROVER DNS resolvers around the world, waiting for the TTL to expire. The new record will not be queried until then. If the operator changes the router AS without publishing a new SRO, the route will be detected as BOGUS. The TTL is creating a classic "damned if you do, damned if you don't" situation.

To address these issues, a new draft `02` of `draft-gersch-grow-revdns-bgp` [13] was submitted on February 25, 2013. The draft defines modifications to the SRO and RLOCK records that incorporate new fields besides the route origin. These include *flag, activation time, and prefix-limit* fields.

The *flag* field is currently unused, but is there to allow for future expansion. The *activation time* field is used to handle experimental publishing and to aid in the transfer of an IP prefix from one AS to another. As we stated in the IETF document:

The purpose of the Activation Time field is to permit publication of SRO records in the reverse DNS prior to its use in formal route validation. This enables two key capabilities: first,it allows for the testing of RLOCK records in a safe manner by informing an application to "don't validate, *but tell me what you would have done". Second, it allows an ISP to publish an RLOCK and SRO record that defines a large covering prefix to give advance warning to that ISP's customers. Customers that have their own AS number and a delegated address block within the ISP's larger block may want to publish their own SRO record if they share a zone with the ISP, or they will risk having their announcements being marked INVALID because the ISP has claimed a larger covering prefix. Alternatively, the customer may be independent from the ISP's zone and manage their own reverse-DNS zone that has been delegated to them. In this case they may choose to publish an SRO record or to do nothing. The cut point of the zone delegation stops the ISP's covering prefix from extending into the new zone.*

We had also considered including an *expiration time* field to indicate when the SRO or RLOCK was no longer valid. In fact, this would be a very useful device for ROVER reliability – the ROVER database would store information for an IP prefix that would potentially be valid for a very long time, perhaps months or years. The problem is that operators make mistakes. It is very likely that no one will change an expiration date even after the record expires. If this were to be implemented, an automatic provisioning system would be needed to refresh the expiration dates. Otherwise some sort of warning mechanism would be needed to alert operators to update their zone files.

In any event, it is easy to remove or modify a record from the reverse-DNS if an operator no longer wishes to publish route origins or is changing delegations. So it isn't really necessary to have an expiration time in the ROVER resource records.

The *prefix-limit* field was added to the SRO record due to an experiment in which ROA records from RPKI were translated to their equivalent SRO form. ROA records implement a *prefix-limit*, and originally it was thought that this would not be necessary for ROVER because a zone cut would stop the effective coverage of an RLOCK statement. This turned out to be impractical for

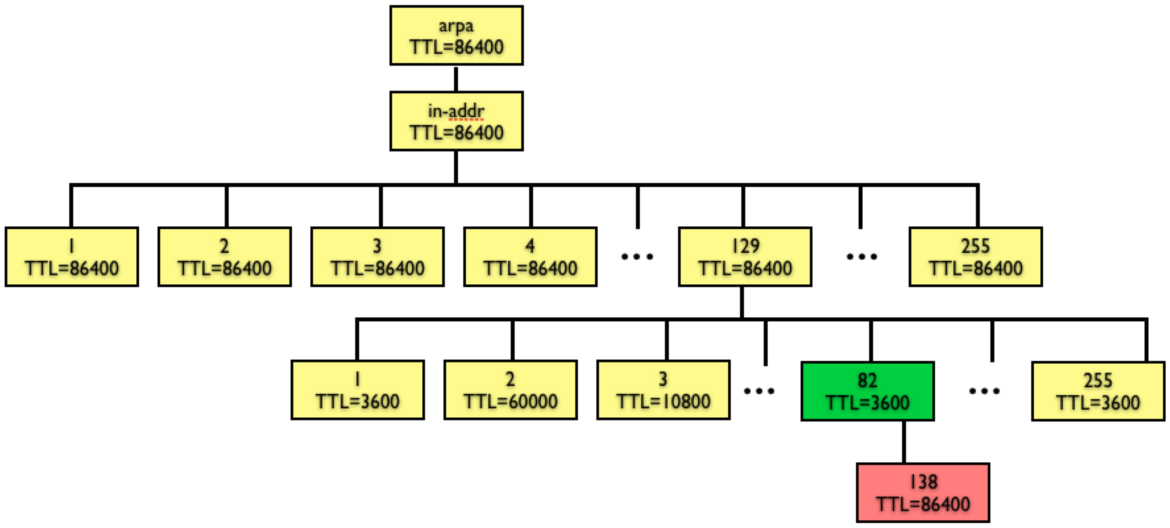


Figure 7.3. *The Reverse DNS tree showing fanout from the root to lower-level authoritative DNS servers*

IPv6, however. Consider an IPv6 /48 prefix that needs to protect all sub-prefixes up to /64 but not beyond. This would require 65,536 NS statements (zone cuts) to be present in the zone, which is clearly impractical. Simply adding a prefix-limit to the SRO statement eliminated this problem.

7.3. *Global DNS Capacity and Load*

Every BGP route announcement will cause ROVER to issue DNS queries for the associated SRO and RLOCK records. Some of these will queries will be answered immediately from the DNS resolver's local cache. If the data is not locally available, then the DNS resolver will issue recursive queries to globally distributed DNS authoritative servers to assemble its response.

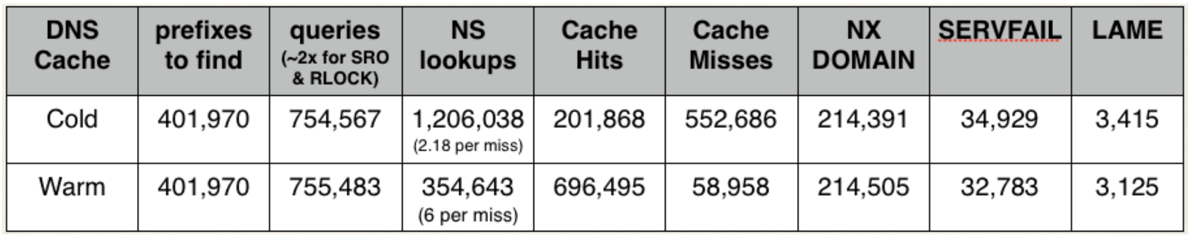
These authoritative servers are a critical dependency for ROVER functionality. But will these servers be able to handle the increased load from ROVER queries? in this section we examine the expected load and compare it to the capacity of these authoritative servers.

Figure 7.3 displays the reverse DNS tree structure and shows how rapidly these names fan out to different zones. Although there are 255 different entries at each level, multiple zones are often managed by one organization. RIPE, for example, owns several blocks of addresses, ARIN owns a different set, and so forth. The same thing occurs at lower levels as multiple ISPs may own several different address blocks. Experiments have shown total fanout of the reverse-DNS tree is shared among approximately 50,000 DNS servers. Note, however, that this does not imply 50,000 separate organizations. Name servers are typically master/slave pairs. RIPE and ARIN have six way redundancy. And even these 6 separate servers per zone file may be backed up with anycasted servers to create additional capacity and redundancy.

Resolving a reverse-DNS name requires multiple queries to these authoritative servers. As an example, the domain name associated with Colorado State University's NETSEC lab is 138.82.129.in-addr.arpa. If nothing were in the DNS resolver's cache, it would take 5 recursive queries to resolve the name, each query traversing the next lower level in the reverse-DNS tree. On the other hand, the first three layers of the DNS tree are typically long-lived in the resolver's cache memory. Therefore a query for the NETSEC lab will usually only require two authoritative queries: one for the 82 subdomain and one for the 138 subdomain.

We now examine a worst-case scenario. Given a freshly-started ROVER device with nothing in its DNS cache, what will happen when it issues 400,000 queries as it iterates through a RIB to pre-load the ROVER data base and DNS cache? How many authoritative queries will be issued, how far will they fan out? Which authoritative servers get the heaviest load?

A Python program was written that performed DNS lookups on a full RIB table. The program performs 400 parallel-threaded queries for each prefix in the RIB. Parallel queries are required because the average response time of a query varies widely, from 30 milliseconds to many seconds, depending on failures, timeouts, retries, SERVFAIL conditions, etc. It is unacceptable to wait for a single query to finish before beginning the next. This analysis represents a naive DNS lookup system. Modern DNS servers can do tens of thousands of parallel queries to name servers.



DNS Cache	prefixes to find	queries (~2x for SRO & RLOCK)	NS lookups	Cache Hits	Cache Misses	NX DOMAIN	SERVFAIL	LAME
Cold	401,970	754,567	1,206,038 (2.18 per miss)	201,868	552,686	214,391	34,929	3,415
Warm	401,970	755,483	354,643 (6 per miss)	696,495	58,958	214,505	32,783	3,125

Figure 7.4. *Results from DNS Query Experiment*

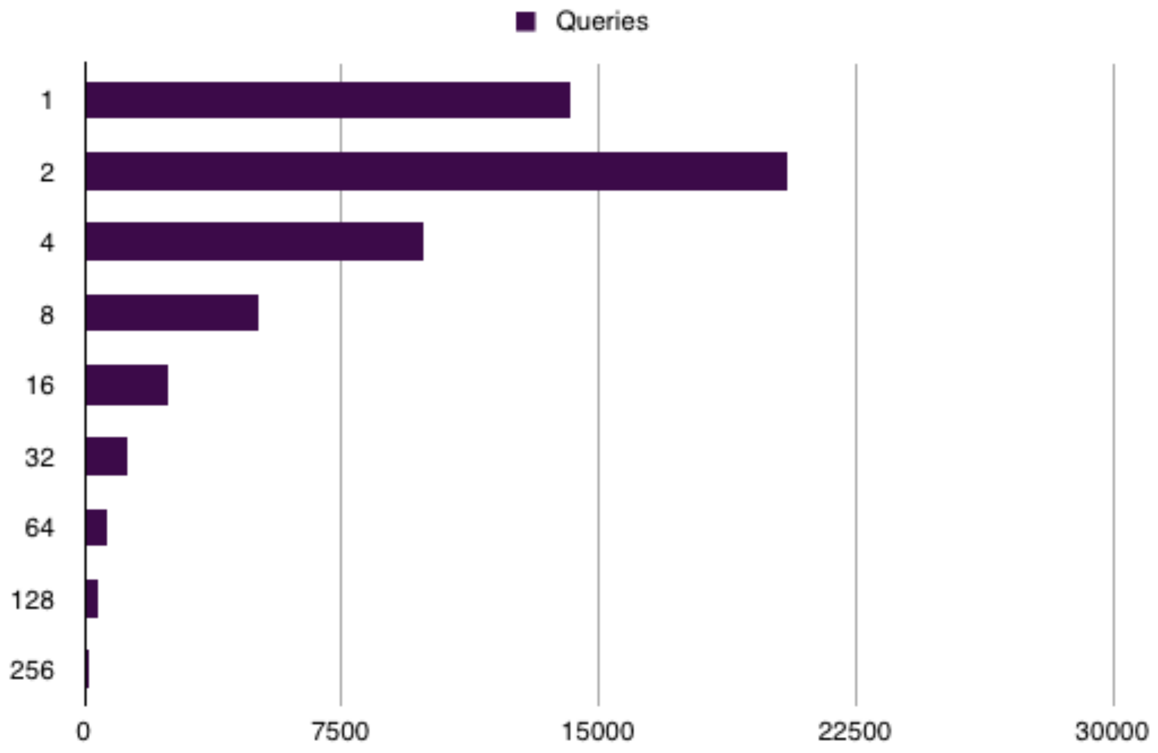
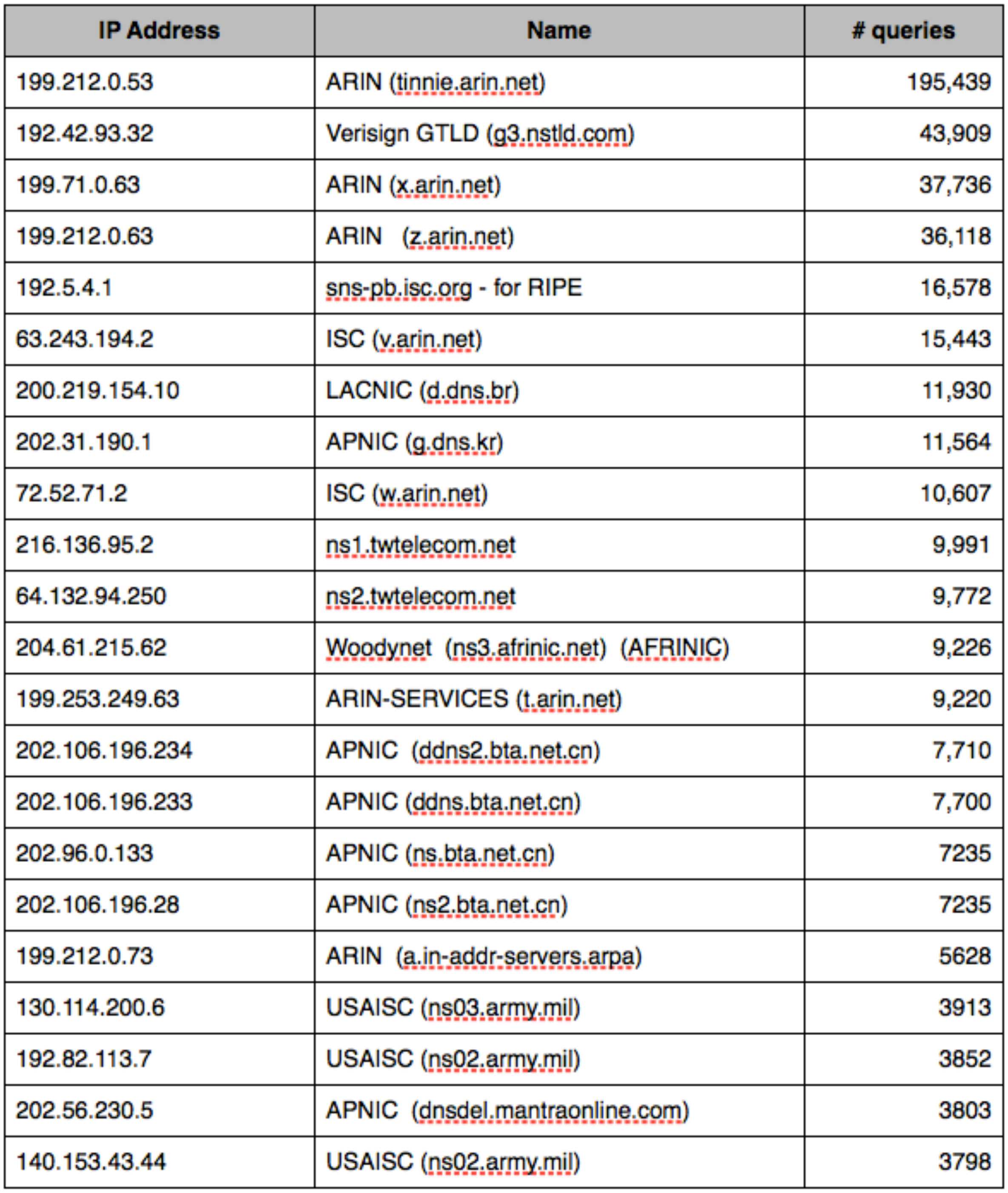


Figure 7.5. *Query Distribution vs # Servers*

The program averaged about 1000 queries per second and completed in 7 minutes. The query load fanned out to 50,000 DNS name servers world-wide, as illustrated in Figures 7.5 and 7.6. As expected, the main servers at RIPE, ARIN, and the other RIR's were the most referenced DNS servers, and then the queries fanned out widely.

Figure 7.4 shows the number of queries generated. The first row shows data for an empty DNS cache (cold start). A second run was performed immediately thereafter (warm start) and is shown on the second row. In each case 401,970 prefixes must be authenticated.



IP Address	Name	# queries
199.212.0.53	ARIN (tinnie.arin.net)	195,439
192.42.93.32	Verisign GTLD (g3.nstld.com)	43,909
199.71.0.63	ARIN (x.arin.net)	37,736
199.212.0.63	ARIN (z.arin.net)	36,118
192.5.4.1	sns-pb.isc.org - for RIPE	16,578
63.243.194.2	ISC (v.arin.net)	15,443
200.219.154.10	LACNIC (d.dns.br)	11,930
202.31.190.1	APNIC (g.dns.kr)	11,564
72.52.71.2	ISC (w.arin.net)	10,607
216.136.95.2	ns1.twtelecom.net	9,991
64.132.94.250	ns2.twtelecom.net	9,772
204.61.215.62	Woodynet (ns3.afrinic.net) (AFRINIC)	9,226
199.253.249.63	ARIN-SERVICES (t.arin.net)	9,220
202.106.196.234	APNIC (ddns2.bta.net.cn)	7,710
202.106.196.233	APNIC (ddns.bta.net.cn)	7,700
202.96.0.133	APNIC (ns.bta.net.cn)	7235
202.106.196.28	APNIC (ns2.bta.net.cn)	7235
199.212.0.73	ARIN (a.in-addr-servers.arpa)	5628
130.114.200.6	USAISC (ns03.army.mil)	3913
192.82.113.7	USAISC (ns02.army.mil)	3852
202.56.230.5	APNIC (dnsdel.mantraonline.com)	3803
140.153.43.44	USAISC (ns02.army.mil)	3798

Figure 7.6. *Table of Most Popular Servers Queried*

In the cold-start case, a total of 754,567 DNS lookups were issued; approximately 2 queries for each prefix, one for the SRO (which may not found) and one for the RLOCK. In the process of looking up a record, one needs to learn the authoritative name servers (NS lookups) for the reverse zones. A total of 1,206,038 authoritative name server lookups were preformed as part of the normal DNS resolution process. Since the cache was cold, one would expect a large number of cache misses (there were 552,686 misses in total). During the lookup process, subdomain names are cached and 201,868 of the recursive domain name queries become cache hits due to previous queries.

The queries usually returned with "domain found, but no SRO record available", or with NXDOMAIN or even SERVFAIL responses. There were a total of 214,391 NX domain responses and 34,929 server fails. 3415 of the name servers learned were lame. Note these results come from the existing reverse DNS and are consistent with past DNS studies.

The second row is similar to the first, except the now the first experimental run has loaded the DNS resolver's cache. The number prefixes to lookup remains the same. The number of queries issued varies slightly due to DNS retries; different queries will try different authoritative servers. Additionally some queries from the python program to the resolver were lost over the network. This result shows experimental evidence for the resiliency improvements discussed earlier.

The number of cache hits increase dramatically due to data being present in the cache. There are some cache misses (58,958) due to cache dynamics. In particular, SERVFAIL responses have expired from the cache. The number of NXDOMAIN, SERVFAIL, and LAME servers remains consistent since both experiments use the same real reverse DNS.

While this may seem like a lot of queries, the DNS system is designed for speed and redundancy. Many of the servers, especially those at RIRs such as RIPE and ARIN, are ANYCASTed to handle large loads. To illustrate, the worst case for a query load to a single name server at ARIN was a total of 195k queries out of the total 1.2 million. Over 420 seconds, this is only 464 queries per second, a very light load for a system designed to handle world-wide DNS queries.

At the other end of the DNS tree, the reverse-queries have fanned out widely. This is evident in Figure 7.6 as it shows the query rate dropping rapidly as the load fans out. It is also evident in Figure 7.5, which shows that approximately 14,000 DNS servers get only one query, and an additional 20,000 DNS servers only get two queries. Therefore we can conclude that ISP's will hardly notice any increased load to their DNS service.

Now assume that 1000 ROVER devices all schedule a RIB iteration at the same time. ARIN would receive 464,000 queries per second. This is significant, however the ARIN DNS servers are provisioned to handle millions of queries per second. The lower-level DNS servers at ISP's and enterprise organizations receive a much lower load, 1000 or 2000 queries over seven minutes. This is easily handled by modern DNS servers.

Finally, it must be noted that this entire scenario happens only infrequently. The normal DNS load will simply be driven by the BGP update rate. Experiments from the ROVER testbed have shown a range from 5 to 600 prefixes processed per second, as these come from the CSU BGPmon service peered with multiple routers. A standalone router would be nominally in the range of 1 to 30 updates a second, with occasional bursts if it needs to process a RIB transfer. These rates are very low compared to the capacity of the DNS. Therefore we conclude that global DNS capacity is not a problem, and even if it were, the DNS is quite capable of scaling larger.

7.4. *ROVER Criticisms and Objections*

ROVER has been presented to global audiences at IETF, NANOG, RIPE, and other venues. This has given us an opportunity to listen and respond to questions and criticisms raised by attendees, primarily those associated with the RPKI effort. Their concerns focus on:

- lack of enumeration,
- a perceived circular dependency between BGP and DNS,

- lack of provenance data,
- potential business relationship gap in certain DNS sub-delegations.

We address each of these in turn.

- **Lack of enumeration:** RPKI proposes a mechanism for an ISP to retrieve collections of ROAs (Route Origin Authorizations) in an offline manner, typically once per day. This mechanism constructs a static dataset of route-origins that can be enumerated and indexed by AS number. A separate protocol transmits this dataset to a router to determine BGP announcement validity.

  ROVER has been criticized for its inability to construct an enumerated list directly from the DNS. This, in general, is a true statement. The DNS does contain a search capability; one must know the exact domain name and RRType to retrieve data. It is impractical, if not impossible, to search the entire reverse-DNS tree to build an enumerated list of SRO records. The RPKI objection is that ROVER cannot therefore work without having a pre-built list of origin authorizations.

  This objection has at least two counter-arguments. The first is that ROVER applications do not necessarily need an enumerated list. Route origins are retrieved *on demand* in near real-time. This allows for much more dynamic data modification. Additionally, ROVER applications cache DNS records and construct a database of route origins learned from ongoing route-announcements. This builds the equivalent of an enumerated list as a complete RIB is traversed.

- **Circular dependency:** Occasionally a question is raised doubting that a low-level protocol, BGP, should rely on a higher-level protocol, DNS. The typical example is a hypothetical restart of the entire Internet. Routers cannot access the DNS during the reboot process since no routes have been established yet. Therefore BGP cannot have a circular dependency on DNS.

In fact, this is true. BGP should not rely on DNS. In the case of ROVER, however, this circular argument does not apply.

The fail-safe mechanism of ROVER requires that routers be allowed to establish routes even when ROVER is not available. Routing will work just as it works now (with no routing authentication) until DNS connectivity is established. At that point routes can be re-checked by iterating through the ROVER database.

Additionally, the ROVER database and DNS cache contain a local storage of ROVER route origins. Even during a global re-boot of the Internet, these data can be referenced for authentication.

Given the fail-safe mechanism and local storage for resiliency, the circular dependency is moot.

- **Provenance:** ROA certificates have a chain of signed signatures from which one can determine the ownership and provenance of a particular address block. For example, one could follow a signature chain from RIPE to ISP-X to ISP-Y to determine who owns an address block and from whom it was assigned or allocated.

  Provenance is, simply put, not a primary objective of the ROVER design. Provenance is not necessary to prove address block ownership. The DNS delegation mechanism is sufficient and is simpler to manage than a PKI infrastructure which must manage certificate revocation and other PKI operational issues.

On the other hand, address block ownership is a desirable attribute, and it is certainly conceivable that future enhancements to ROVER could accommodate DNS records which contain address block ownership information.

- **DNS sub-delegation:** An objection was raised regarding certain DNS delegation chains when certain address blocks are sub-divided. As an example, organization A could split its /20 into two /21 blocks and assign one of them to organization B. Later, organization B could sell part of its space to organization C. If the delegations cross an octet boundary, organization C must send a DNSSEC DS record to be stored in the zone owned by organization A, not to organization B. The problem is that organization C may only have a business relationship with organization B. If organization A receives a DS record from organization C and does not have a business relationship with C, they may not accept the DS record. This breaks the DNSSEC chain of trust.

  Simply put, the situation just described is not a ROVER issue. This business relationship issue is known to occasionally occur as a consequence of basic DNSSEC delegation operations, with or without ROVER. The problem can be resolved by simply establishing a business relationship from organization C to A when the delegation is first made by organization B.

7.5. *Conclusion*

ROVER uses the reverse-DNS as a global distribution mechanism to publish and retrieve route origin data. The basic properties of the global DNS create a compelling reason for its use. It scales, it is performant, it is resilient, allows for dynamic updates, and has been proven in public use for decades.

Despite these advantages, the DNS is subject to possible communication failure or attack. The reverse-DNS by itself cannot be completely relied upon to alway return a response. For that matter, no Internet publication method can be deemed entirely reliable.

To compensate for this, ROVER combines the use of the reverse DNS with several failure recovery mechanisms. The DNS cache and the ROVER database are available for immediate local lookup of ROVER data. Exponentially timed re-queries attempt to retrieve data. In the event that a query fails, the historical transactions contained in the ROVER database are consulted. If the historical SRO record matches the BGP prefix announcement, it is allowed to pass, but marked INSECURE. A log entry and operator notification is issued indicating whether the BGP announcement matches the historical ROVER data; if not, the operator receives a warning message.

The end result of combining DNS data retrieval, local storage, and operator notifications is a resilient system with significant reliability. As a final note, we recall the ROVER design objective to fail safely. Even if the ROVER system is entirely disabled, the global BGP system will continue to operate just as before, but without the benefit of authenticating route origins.

CHAPTER 8

CONCLUSION

This dissertation introduced the ROVER (Route Origin VERification) approach for detecting origin and sub-prefix route hijacks. ROVER provides a mechanism to securely publish authoritative route origins in the existing reverse DNS infrastructure. To accomplish this, ROVER introduces a naming convention that provides a unique DNS name for an IPv4 and IPv6 prefix. Network operators authorize the prefix to be announced and specify an origin AS by placing a Secure Route Origin (SRO) record at the prefixs DNS name. An additional RLOCK record is used to indicate whether the prefix is using the ROVER publishing approach. This published data may then be used in a variety of applications to detect or prevent IP hijacks.

This dissertation also presented an analysis and quantification of the global internet topology with respect to IP Hijack propagation. Two new metrics were introduced, *depth* and *reach*. These were shown to correlate well with the extent of an IP hijack. We also presented an analysis of the effectiveness of incremental rollout of BGP security mechanisms. An empirical analysis using simulation enabled us to characterize AS vulnerability to route origin hijacks. Wide variance of attack resistance and attack aggressiveness was observed. This variance most strongly correlated with the depth of an AS. We then demonstrated that prevention and detection technology can be effective with incremental rollout. The ability to block IP hijacks increases non-linearly when more blocking mechanisms are deployed, however a small critical mass is required to enable a reasonable level of protection. We also demonstrated that hijack detection can be highly effective, but that once again a critical mass of probes must be present to avoid blind spots, Finally, we proposed a set of short-term actions that could be taken to initiate incremental BGP security rollout.

This approach would have immediate benefits to self- interested AS organizations and help build towards the critical mass needed in general.

Finally, this dissertation presented an analysis of ROVER with respect to its external dependencies, and potential weakness and vulnerabilities that could be exploited by an attacker. The critical dependence on the reverse-DNS was analyzed regarding reliability, capacity and scalability. Operational considerations revealed design shortcomings. These studies resulted in architectural improvements that make the ROVER system robust, resilient and reliable.

8.1. *Contributions*

Novel work is as follows:

(1) The survey of IP hijack detection and prevention papers led to the development of the solution taxonomy given in Section 2.3. The economic arguments pitting ASes with high risk versus the majority of apathetic ASes was fundamental to the argument for incremental deployment.

(2) ROVER: the design of the ROVER naming convention which allows delegation, the evolution of the resource record extensions, and the development of a working, robust ROVER validation algorithm are contributions to the overall ROVER architecture.

(3) the Web-based ROVER TESTBED allowed world-wide users to experiment with ROVER concepts and view real-time ROVER hijack detection as a working application. Additionally, the testbed module for publishing zone files is a unique contribution due to the manner in which brings up a tree structure of address blocks and previously seen AS announcements. As this testbed evolved it helped us understand the shortcomings in the validation algorithms and resource records as well as identifying resilience and reliability issues. This testbed has the potential for commercialization.

(4) The Global Internet Simulation program, its underlying database of AS information, and the graphics programs for producing polar charts and Hilbert maps were indispensable in the analyses performed. We believe these tools could be commercialized for use by organizations who wish to analyze their own attack vulnerability and implement self-interested defenses. Of course, the tools could also be used to craft more sophisticated attacks as well.

(5) The research results regarding Internet measurements, vulnerability analysis and incremental defense deployment defenses are new contributions. The *depth* and *reach* metrics are introduced in this dissertation. Although one could argue that these are nothing more than indicators of path length, but in fact they provide a novel way to determine AS vulnerability through metrics associated with the AS position in the Internet topology.

(6) The analysis method regarding ROVER robustness and resilience demonstrates a process for dealing with a system which starts out with fundamental reliability issues and shows how it becomes more robust by analyzing weaknesses, threat models and operational considerations.

8.2. *Future Directions*

There are many fruitful areas of research, development and refinement than can be pursued in the future. Among these are:

- Numerous experiments and more involved statistical analyses could be proposed to develop a deeper understanding of attack vulnerability and defensive measures. Future work is required to understand the behavior of the internet topology with respect to the holes still present in an incremental deployment. Even with a deployment of critical mass ROVER devices, some origin and sub-prefix attacks will still get through, and possibly remain undetected. An analysis is desirable to understand these attacks, to determine how they remain invisible, and what can be done short of complete global deployment of BGP security at every AS.

- Origin and sub-prefix hijacks are the most common IP hijacks, but as pointed out in chapter 2 there are certainly other issues to resolve. There are ideas worth pursuing regarding other IP hijacks including techniques for next-hop validation, full path validation, and routing valley detection.
- Determine methods for a ROVER application to communicate directly with routers to enable real-time authenticated hijack filtering. It may be possible to use the RTR protocol (RFC6810: RPKI to Router Protocol). It may be possible to replace bogus announcements by having the ROVER application announce a better route, or some other technique may need to be proposed.
- One of the the most fruitful areas for future research would be in the area of "pragmatic self-interest" discussed in Section 6.3. How does an ISP ensure that its customers will not be hijacked? A process involving AS topology mapping, vulnerability analysis, simulation, and defense strategies could be a practical outcome.
- Finally, the actual effort to get ROVER deployed and accepted as a world-wide ad hoc or de jury standard is the most important direction to pursue, proving its worth beyond its academic contribution.

CHAPTER 9

PUBLICATIONS AND PRESENTATIONS

This work has resulted in a combination of publications in IEEE academic conferences, peer reviewed presentations, operational venues such as the North American Network Operators Group (NANOG) and other similar regional groups, preliminary standards discussions in the Internet Engineering Task Force (IETF), and international attention focused on ROVER by government organizations, tier-1 ISPs, enterprise organizations and others.

ROVER has been presented to the ISOC Outreach Committee (Internet Society), the U.S. Federal Communication Commission CSRIC Working Groups (Communications Security, Reliability and Interoperability Council) where J. Gersch is a member of CSRIC working group 4 and C. Garner of Century Link is a member of CSRIC working group 6. Discussions on implementing and evolving ROVER have been held with Century Link and Level3 over the last year. M. Glenn of Century Link has presented ROVER to their government customers as well as other conferences.

Interest in ROVER is also indicated by the organizations that have signed up for the ROVER experimental testbed including Hewlett-Packard, ATT, Century Link, Internet2 members, and others. This level of global interest demonstrates the impact the ROVER technology is creating and its potential for future deployment either as a *de jure* or *de facto* future standard.

(1) J. Gersch and D. Massey. “Reverse-DNS naming convention for CIDR address blocks”, In Securing and Trusting Internet Names - SATIN 2012, March 2012.

(2) J. Gersch, D. Massey, and E. Osterweil. “Reverse DNS Naming Convention for CIDR Address Blocks. draft, IETF”, February 2012. Revised May 2012. Revised October 2012. Revised February 2013. http://tools.ietf.org/html/draft-gersch-dnsop-revdns-cidr.

(3) J. Gersch, D. Massey, C. Olschanowsky, and L. Zhang. "DNS Resource Records for BGP Routing Data, draft, IETF", February 2012. Revised August 2012. Revised February 2013. http://tools.ietf.org/html/draft-gersch-grow-revdns-bgp.

(4) J. Gersch, D. Massey. "IETF draft documents". presentations of the two draft documents listed above at IETF83 *grow* and *dnsop* working groups, March 2012, Paris, France.

(5) J. Gersch. "Secure64 ROVER web site and testbed". March 2012, continuously revised. http://rover.secure64.com.

(6) J. Gersch and D. Massey. "BGP Route Origin Verification Using Reverse DNS", presentation at RIPE64 Conference, April 2012, Ljubljana, Slovenia.

(7) J. Gersch, D. Massey, M. Glenn, and C. Garner. "ROVER: BGP route origin verification via DNS". presentation at NANOG 55, June 2012, Vancouver, BC, Canada.

(8) D. Massey and J. Gersch. "BGP Route Origin Verification Through DNS", presentation at AUSNOG Conference, September 2012, Melbourne, Australia.

(9) D. Massey and J. Gersch. "Monitoring and Protecting Internet Routes with BGPmon and ROVER", presentation and tutorial at TIP2013 (joint conference of APAN, Internet2 and ESnet), January 2013, Honolulu, Hawaii

(10) J. Gersch and D. Massey. "ROVER: Route Origin Verification Using DNS". In ICCCN 2013: International Conference on Computer Communications and Networks, July 2013.

(11) J. Gersch and D. Massey. "Characterizing Vulnerability to IP Hijack Attempts". In IEEE-HST 13: IEEE International Conference on Technologies for Homeland Security, November 2013.

(12) J. Gersch, D. Massey and C. Papadopoulos. "Effective Incremental Deployment of BGP Security", submitted to IEEE INFOCOM 2014.

BIBLIOGRAPHY

[1] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the internet. In *Proceedings of ACM SIGCOMM*, August 2007.

[2] T. Bates, R. Bush, T. Li, and Y. Rhekter. DNS-based NLRI origin AS verification in BGP. draft, IETF, July 1998. http://tools.ietf.org/html/draft-bates-bgp4-nlri-orig-verif-00.

[3] P. Booth, J. Hiebert, and R. Bush. Short-lived prefix hijacking on the Internet. In *NANOG 36*, February 2006.

[4] M. Brown. Renesys Blog. Pakistan Hijacks YouTube. http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml.

[5] K. Butler, T. Farley, P. McDaniel, and J. Rexford. A survey of bgp security issues and solutions. In *Proceedings of the IEEE*, volume 98, pages 100–122, January 2010.

[6] CAIDA. AS Relationships. http://www.caida.org/data/active/as-relationships/index.xml.

[7] CAIDA. Visualizing IPv4 and IPv6 Internet Topology at a Macroscopic Scale in 2011. http://www.caida.org/research/topology/as_core_network/.

[8] L Gao. On inferring autonomous system relationships in the Iinternet. *IEEE/ACM Trans. Networking*, 9(6):773–745, December 2001.

[9] J. Gersch. Secure64 ROVER web site and testbed. http://rover.secure64.com.

[10] J. Gersch and D. Massey. Reverse-DNS naming convention for CIDR address blocks . In *Securing and Trusting Internet Names - SATIN 2012*, March 2012.

[11] J. Gersch and D. Massey. Characterizing Vulnerability to IP Hijack Attempts. In *IEEE-HST: IEEE International Conference on Technologies for Homeland Security*, November 2013.

[12] J. Gersch and D. Massey. ROVER: Route Origin Verification Using DNS. In *ICCCN 2013: International Conference on Computer Communications and Networks*, July 2013.

[13] J. Gersch, D. Massey, C. Olschanowsky, and L. Zhang. DNS Resource Records for BGP Routing Data. draft, IETF, February 2012. http://tools.ietf.org/html/draft-gersch-grow-revdns-bgp-01.

[14] J. Gersch, D. Massey, and E Osterweil. Reverse DNS Naming Convention for CIDR Address Blocks. draft, IETF, February 2012. http://tools.ietf.org/html/draft-gersch-dnsop-revdns-cidr-03.

[15] P. Gill, M. Schapira, and S. Goldberg and. Let the market drive deployment: a strategy for transitioning to BGP security. In *Proceedings of the ACM SIGCOMM 2011 conference*, 2011.

[16] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How Secure are Secure Interdomain Routing Protocols? In *SIGCOMM '10: Proceedings of the ACM SIGCOMM 2010 Conference on Data Communication*, June 2010.

[17] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: an incremental approach to improving security and accuracy of inter domain routing. In *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS 03)*, February 2003.

[18] C. Hall. Security of the internet and the known unknowns. *Communications of the ACM*, 55(6):35–37, June 2012.

[19] C. Hepner and E. Zmijewski. Defending against BGP man-in-the-middle attacks. In *Blackhat DC '09*, February 2009.

[20] P. Hoffman and J. Schlyterl. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS Protocol: TLSA. Rfc, IETF, August 2012. http://tools.ietf.org/html/draft-ietf-dane-protocol-23.

[21] C. Horn. Understanding IP prefix hijacking and its detection. In *Seminar internet routing, intelligent networks (INET)*, 2009.

[22] X. Hu and Z. M. Mao. Accurate real-time identification of ip prefix hijacking. In *Proceedings of IEEE Symp. Security and Privacy*, May 2007.

[23] G Huston and R Bush. Securing BGP with BGPsec. *The ISP Column*, July 2011.

[24] G. Huston, M. Rossi, and G. Armitage. Securing BGP - a literature survey. *IEEE Communications Surveys and Tutorials*, 2011.

[25] J Karlin, S Forrest, and J Rexford. Pretty good BGP: improving BGP by cautiously adopting routes. In *Proceedings of IEEE Int'l Conf on Network Protocols (ICNP)*, 2006.

[26] E. Katz-Basset, D. Choffnes, I. Cunha, C. Scott, T. Anderson, and A. Krishnamurthy. Machiavellian routing: improving internet availability with BGP poisoning. In *HotNets-X: Proceedings of 10th ACM workshop on Hot Topics in Networks*, November 2011.

[27] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 75–86, October 2010.

[28] M. Lad, D. Massey, D. Pei, Y. Wu, and B. Zhang. PHAS: a prefix hijack alert system. In *Proceedings of 15th USENIX Security Symposium*, August 2006.

[29] PC Magazine. ISPs Agree to FCC Rules on Anti-Botnet, DNSSEC, Internet Routing. http://securitywatch.pcmag.com/security/295722-isps-agree-to-fcc-rules-on-anti-botnet-dnssec-internet-routing.

[30] MAXMIND. Geolocation Information. http://www.maxmind.com.

[31] C. McArthur and M. Guirguis. Stealthy IP prefix hijacking: Don't bite off more than you can chew. In *Proceedings of the 28th IEEE conference on Global telecommunications*, November 2009.

[32] P. McDaniel, W. Aiello, K. Butler, and J. Ioannidis. Origin authentication in interdomain routing. *Computer Networks*, 50(16):2953–2980, November 2006.

[33] L. Neudorfer, Y. Shavitt, and N. Zilberman. Improving AS Relationship Inference Using PoPs. In *IEEE INFOCOM 2013 International Traffic Monitoring and Analysis Workshop*, April 2013.

[34] NLNetLabs. BGP Model and Simulation. http://www.nlnetlabs.nl/projects/bgpsim.

[35] O. Nordström and C. Dovrolis. Beware of BGP attacks. *Computer Communications Review*, 34(2):1–8, 2004.

[36] University of Arizona Network Research Lab. Large Route Leaks. http://dyadis.cs.arizona.edu/projects/lsrl-events-from-2003-to-2009.

[37] U.S. Department of Commerce. U.S. Census Bureau News, Quarterly Retail E-Commerce Sales, 2nd Quarter 2013. http://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf.

[38] University of Oregon. Route Views Project. http://www.routeviews.org.

[39] T. Paseka. Why Google went offline today. http://blog.cloudflare.com/why-google-went-offline-today-and-a-bit-about.

[40] T. Qiu, L. Ji, D. Pei, J.Wang, J. Xu, and H. Ballani. Locating prefix hijackers using LOCK. In *Proceedings of 18th Conference on USENIX Security Symposium*, 2009.

[41] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proceedings of ACM SIGCOMM*, August 2006.

[42] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). rfc 4271, IETF, January 2006.

[43] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J.Wu. Detecting prefix hijackings in the Internet with Argus. In *Proceedings of the 2012 ACM Internet Measurement Conference (IMC'12)*, November 2012.

[44] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: security mechanisms for BGP. In *Proceedings of Symp. Networked Systems Design and Implementation (NSDI)*, March 2004.

[45] J. Taylor. How did Dodo break the internet. *ZDNet*, February 2012.

[46] A. Toonk. BGPmon.net: a BGP monitoring and analyzer tool. http://bgpmon.net.

[47] UCLA. Cyclops: a network audit tool. http://cyclops.cs.ucla.edu.

[48] Colorado State University. BGPmon: Next generation BGP Monitor. http://bgpmon.netsec.colostate.edu.

[49] M. Wählisch, O. Maennel, and T. Schmidt. Toward detecting BGP route hijacking using the RPKI. In *Proceedings of Sigcomm '12*, August 2012.

[50] Wikipedia. IT risk management. http://en.wikipedia.org/wiki/IT_risk_management.

[51] Z. Zhang, Y. Zhang, Y.C. Hu, and Z.M Mao. Practical defenses against BGP hijacking. In *Proceedings of ACM CoNext Conference*, 2007.

[52] Z. Zhang, Y. Zhang, Y.C. Hu, Z.M. Mao, and R. Bush. ISPY: detecting IP prefix hijacking on m own. *IEEE/ACM Transactions on Networking (TON)*, 2010.

[53] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. An analysis of BGP multiple origin AS (MOAS). In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[54] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis. A lightweight distributed scheme for detecting IP prefix hijacking in realtime. In *Proceedings of ACM SIGCOMM*, 2007.