Detecting inconsistencies in INRDB data

to identify MOAS cases and possible illegitimate internet resource usage

ing. P.Ruissen

System and Network Engineering

University of Amsterdam

December 10, 2007

Abstract

RIPE NCC has a new experimental Internet Number Resource database (INRDB). The INRDB holds statistic data about address space allocation, routing registry information and other related objects. The assignment in this project is to use historical resource data to correlate inconsistencies between INRDB data sources to identify MOAS cases and detect possible illegitimate internet resource usage. The results show the RIS RIB table growth of one 62/8 from 1495 unique prefixes in 2004 to 2166 unique prefixes in 2007. It also shows that the average of 88% RIS entries that match AS_ORIGINS in RIPEDB exceeds expectations. The MOAS results of the implementation of the algorithm (which are not listed here) showed that almost 80% percent of all MOAS keeps coming back every month. However, correlating these MOAS cases with listings in RIPEDB and registration data is not enough to determine if they are hijacked or not. For this we need overlap detection, timeframe processing and resource certification validation.

Contents

Listings

PREFACE

This report is written for RP1 as part of the MSc study in the field of SNE at the UVA. The timeframe is one month and the goal is to gain practical experience and collaboration skills. During this project I was invited to work as a trainee within the Science Group of RIPE NCC and to attend the RIPE55 and NLUUG meetings. The Science Group provides a scientific analysis and publication resource which is independent of day-to-day operations of the company[21]. The research was performed on behalf of RIPE NCC, Amsterdam under supervision of Daniel Karrenberg and Tiziana Refice. I want to thank them both for their enthusiasm and excellent guidance during this project. I also want to thank Erik Romijn and Robert Kisteleki for their expertise and useful advice. Further, I want to thank RIPE NCC and dr.ir. C.Th.A.M. de Laat from the Faculty of Science for making this research possible.

ABBREVIATIONS

Allocation distribute address space to internet registries for the purpose of distribution by them.
Assign means delegate address space to an ISP or End User for specific use within the AS they operate.
AS stands for autonomous system, a domain of administrative authority.
ASN autonomous system numbers are represented as 2 bytes to identify an AS (0-65535)
AS Path is a sequence of AS numbers traversed from origin AS.
AS Origin The origin AS announces a prefix.
Blackholing attacker drops rerouted packets.
Bogus ASNs are marked as ”Reserved”, ”Held” or ”Designated for private use” by IANA.
Control Plane OSI layer 3 routing info, example: RIS project; AS level paths to each prefix.
Data Plane OSI layer 2, example: IPlane project. [1]; daily traceroutes to prefixes.
Dark address space accessible from one provider but unreachable from competitor networks.
ERX Early Registration Transfer Project; internet numbers allocated before the RIR where established.
Imposture attacker mimics the behavior of target prefix.
Interception MiM attack, attacker forwards hijacked traffic to target prefix.
Ingress Access List filter that checks the source IP of every BGP message against a list of acceptable prefixes.
Inet Number AS number or prefix range (tuple).
INRDB Internet Number Resources database provides info about number resources from a single point.
LIR Local Internet Registries are mostly Internet Service Providers (ISP).
MOAS Multiple origin AS that announce the same prefix.
Multihomed AS maintains connections to more than one other AS. (multiple uplinks).
PA-addresses Provider Aggregated addresses are assigned from an LIR’s allocation.
Peering create a direct link between two AS and exchange local traffic.
PI-addresses Provider Independent addresses are assigned directly by RIR to an end-user without LIR.
Prefix IP network/subnet that represents a single entry in the BGP RIB.
Radix trees Patricia trie/trees, or crit bit trees are specialized set data structures.
RADB Static database similar to RIPEDB.
RIB Routing Information Base contains routes from static/dynamic protocols and directly attached networks.
RIS Routing information service, is a RIPE NCC project to collect and store Internet routing data.
RIR Regional Internet Registry, RIPE NCC is one of the five regional internet registries in the world.
SUBMOAS Subnet of an existing prefix is announced by a different (multiple) origin AS.
Stub AS only connected to one other AS (used in financial, transportation sectors).
Transit AS provides connections through itself to separate networks.

[1] iPlane performs traceroutes from several vantage points daily to map the Internet’s topology. `http://iplane.cs.washington.edu/`

1 INTRODUCTION

There are many sources on the Internet where you can find information about routing policies, registered IP space and ASN, routing table information, registered domains and contact info. One of them is the RIPE whois database, which is part of a global system known as the **Internet Routing Registry**. Other information sources include the RIS (Routing Information Service), ERX, various mail archives and name server query logs.

However, getting an overview of all those information tends to be non-trivial. Users have to query each source to get the complete picture, and up to now there is no single service that combines all this data. For this reason, RIPE NCC started the **Internet Number Resource Database** (INRDB). The INRDB combines various archives into one to create an overview of as much time-series and historical data. This historical resource data can be processed and used to correlate with scientific data, measurements and to address various problems. This research is about the using the combined data resources of the experimental INRDB from RIPE NCC and correlating inconsistencies. The historical data of different sources can be compared and analyzed to possibly detect prefix hijacking.

Hijacked prefixes are address blocks that are used without permissions from their rightful owners, which is considered to be illegal. Mostly this involves duplicate IP addresses announcements or prefixes that are not used somewhere else. Those can also be prefixes from which organizations are not aware that they own them or old blocks that are not in active use on the Internet. Non-legitimately used address-space can then be sold or leased, blackholed, used for deception, denial of service attacks or spamming. Prefix hijacking makes it very difficult to backtrace spam to their original source.

The above information leads to the following **research question**:

- How to correlate inconsistencies between INRDB data sources to identify MOAS cases and detect possible illegitimate internet resource usage? How is the data in the INRDB structured, represented and retrieved? What are the anomalies in this historical data and do we see a trends in malicious activity?

Approach The first phase is to study the INRDB architecture, storage formats and processes. The research topic is covered by writing a Perl script that queries the INRDB data sources, compares RIPEDB, RIS and registry data and relates information. Section 4 covers the properties of hijacked prefixes by using inconsistencies in the INRDB. Comparing datasources is not a trivial task, it involves parsing classes, properties and multidimensional timeframes and writing complex data structures. Results are plotted with the GD::Graph Perl module.

Related work The abundance of research papers about this topic show that there is already a lot of work done in this field. When v4 of the Border Gateway Protocol (BGP) took over in 1994, it finally supported route aggregation which was a great enhancement over previous versions. Unfortunately security was not considered to be an important issue. Since then, a lot of papers where released discussing the security and robustness problems which BGP faced. Misconfiguration was common, some routers could easily flood parts of the Internet with false advertisements [9]. However, some announcements like multiple origin AS may appear bogus, but are actually legimate. [1] For example, the organization may use multihoming or private links.

The fact that malicious activity could also target whole organizations was proven in 2005 when an AS originated routes of one of the prefixes assigned to Google. The result was that Google was unreachable for an hour from a broad range of geographical locations. More can be read in 'Analysis of BGP Prefix Origins During Googles May 2005 Outage'[8]. The range of those attacks can also spread world-wide if ASes higher in the routing topology (tier 1) are hijacked [6].

After the first attacks became apparent, addons were designed to solve the insufficiencies in BGP. Bolt, Beranek, and Newman (BBN) were the first to enhance the BGP protocol with the secure BGP architecture[25] to verify the authenticity and authorization of BGP traffic[17]. CISCO came up with their own solution [24]. Both are discussed in section 1.1. Due to the deployment costs of these solutions many ISP stick with passive measures.. Also, the growth of the RIB routing tables do not make it preferable to deploy cryptography, which uses a lot of resources. This caused that prefix hijacking is still a serious problem to be reckoned with today.

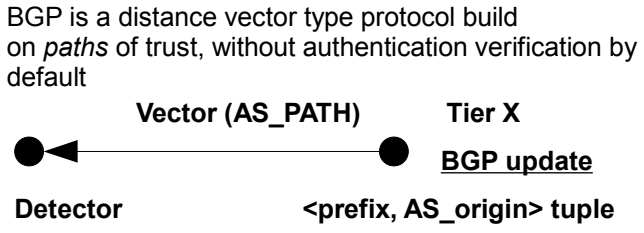
Passive solutions to detect misconfiguration, bogus routes and prefix hijacking can be divided in two categories:

- The first category uses properties from the BGP control plane by monitoring BGP updates and status information. This input information is in almost every case captured from the Route Views project or from RIS. The Prefix Hijack Alert System (PHAS) [16] and MyASN are good examples of projects which alert system administrators when their BGP Origin changes. Other papers present more complicated methods to detect hijacked prefixes like address expansion, neighboring and address sharing heuristics [10].
- The second category use characteristics from the data plane like hop count properties, OS, IP and ICMP timestamp based fingerprints. Other ideas are to use customer provider relations and geographical information.

Scope Due to the limited timeframe and to produce original results, the author has chosen to focus on inconsistencies between data sources, meaning that related information in RIS, RIPEDB and other data sources are compared rather then focusing on inter-dynamic data. However, some papers [2] claim that the RIPEDB WHOIS data is not consistent with the route announcements in RIS and lacks the quality for reliable measurements. Observation and analysis in 5 shows that the historical data in RIPEDB is quite accurate and that it's very suitable for statistics. RIPE NCC is also continuously improving the quality of RIPEDB with policies and scripts.

1.1 BGP SECURITY

The Internet consists out of worldwide interconnected autonomous systems (AS), mostly ISP using BGPv4 RFC4271[20] as the core routing protocol. An AS is a collection of networks with common routing policies and administration identified by a unique autonomous system number (ASN). BGP is a path vector protocol, meaning that each BGP router has to maintain a huge list of IP networks (prefixes) and peers in their routing information base (RIB).



The term path vector is derived from the fact that BGP routing information carries a sequence of AS that a network prefix (CIDR notation) has traversed. A BGP Session starts with the following messages:

- Open message: sets session between peers after a TCP connection is established and confirmed with a keepalive message. It contains the BGP version number, the AS number of sender (called AS_ORIGIN last AS in the path), hold down value (3x keepalive in sec.) and the BGP identifier (IP of BGP speaker).
- Update messages are the most important messages in BGP. They provide reachability info about (un)feasable routes to other BGP systems for a consistent view of the network topology. Each update message also carries info about **path attributes** and NLRI except for messages with only infeasible routes (prefixes). The path attributes define the best path to a destination. The EBGP relevant ones are:
  - The multi-exit discriminator (MED) or metric attribute is used as a suggestion to an external AS regarding the preferred route into the AS that is advertising the metric
  - Origin Attribute: indicates how BGP learned about a particular route. These are on order: Local routes, OSPF, RIP and IBGP
  - Shortest AS path: ordered list of AS that the route advertisement has traversed.
  - Next Hop attribute: is the IP address of the connection between the peers (EBGP) from the advertising BGP router
- Keepalive messages are sent to determine if peers are reachable and keeps the hold down time from expiring. If the router declares the peering session dead; the routes are placed into a dampening state and the session is reset.
- Notification headers are sent when there are errors. The peering session is then closed.

SECURITY MEASURES The current BGP protocol is vulnerable because it relies on a sufficient level of trust between peering AS. The standard version of BGP also lacks means of verifying authenticity and authorization of BGP traffic. Here are some measures to address these problems:

- Using route filtering such as Ingress Access lists to verify the origin AS. However, the downside is that changes in the topology may block legitimate routes or pass fake routes.
- Setting the TTL value to a value to do 'one hop' from the sender. This is known as the 'BGP TTL Security Hack' and works only for adjacent routers and not for multi-hop BGP sessions
- Protect BGP sessions with a MD5 signature hash. Applying MD5 is good for integrity but not more than that.

There are also approaches to fix the security limitations in BGP with cryptographic means:

- Secure BGP (S-BGP) and employs three security schemes. The first mechanism is a x509 public key infrastructure (PKI) to support authentication and authorization of Internet numbers and routers. Second, an optional BGP path attribute is employed to carry digital signatures from BGP updates. The receiver can then verify the signatures of BGP updates with the PKI certificates. The RIR is the **centralized** trusted party that issue these certificates while allocating number resources. The last part is using IPsec to provide data integrity and authenticate BGP routers before exchanging BGP traffic.
- Secure Origin BGP (soBGP) also uses a PKI but is a more lightweight alternative next to SBGP proposed by Cisco. It differs from S-BGP in the way that it does not involve RIR to operate. SoBGP uses a **decentralized** Web of Trust model relying on the existing relations between ISP. SoBGP uses three certificate types. The Entity Certificate (EntityCert) ties an ASN to a key pair. This certificate is signed by a trusted third party like Verisign. The authorization certificate ties an AS to the prefixes it's allowed to advertise. The Policy Certificate (PolicyCert) describes policies related to prefix blocks and the connections between advertising AS and peers.
- Pretty Secure BGP (psBGP) uses a centralized single level PKI for AS number authentication and decentralized trust model for prefix authentication and a rating based approach for path integrity.

- X509 Resource certification (RFC 3280) is a less ambitious project to verify and authenticate usage of number resources (RFC 3779) without AS-path verification. This only requires a centralized PKI hierarchy of public certificates (which contains public key and contact information). Resource certificates follow the same flow as IP resource distribution. The top layer certificates are issued by IANA to the Internet registries and the registries issue them to LIR. The certificates can be used with access lists to block any non-authenticated routes.

We can conclude that there are many cryptographic solutions for securing BGP. But we also know that there aren't many ISP in the real world which actually deploy these solutions. A major obstacle of deploying S-BGP or certification is that it requires participation of all RIR, vendors and ISP to work sufficiently. S-BGP is also expensive to deploy (complex PKI), needs more CPU/memory, requires router upgrades and does not support route withdrawals. Resource certification offers a more intermediate solution, but even that may not be enough. Before LIR even consider resource certification there must be OpenSSL tools in place that are simple, well tested, beneficial and worth the 'hassle'. In other worths, the world just needs one major proven BGP attack to convince large organizations to use resource certification.

1.2 RIPE NCC DATA SOURCES

The Reseaux IP Europeens Coördination Center (RIPE NCC) is a collaborative non-profit organization that consists of European Internet service providers. It aims to provide the necessary administration and coordination to enable the operation of the European Internet[21]. To ensure fair and neutral delegation of Internet Number Resources (ASN and IP addresses) a non-profit organization named ICANN was created in 1998.

ICANN (Internet Coorporation for assigned names and numbers) operates IANA, the entity that oversees global allocation of IP addresses, ASN and domain names. IANA delegates local registration to five regional internet registries, which allocate addresses for a specific part of the world. RIPE is one of the five Regional Internet Registries (RIR) that falls under the Internet Assigned Numbers Authority (IANA). The other RIR are AfriNIC, APNIC, ARIN and LACNIC.

RIPE supports the infrastructure of the Internet and provides global resources (ASN and IPv4/6 addresses) and other services to members within its region. Members which received IP address allocation and assign parts of these allocation to customers are generally referred as Local Internet Registries (LIR). Most of these are Internet Service Providers (ISP) with provider aggregated (PA) address space. Address space which is directly assigned to end users is called Provider Independent (PI). Other responsibilities and activities include management of the K-root server, support for ENUM delegations and providing neutral and public accessible statistics on the operation of the Internet.

RIPE Whois Database RIPE NCC provides development and maintenance of the RIPE (whois) Database which contains registration details about IP addresses and AS numbers. The RIPE Routing Registry is part of this database and contains routing policy information (AS objects) described by the routing policy specification language (RPSL). Routing registries of different RIR are mirrored and form the Internet Routing Registry (IRR). The IRR has long been used to look up peering agreements or determine optimal policies and even to configure routers.

Although the IRR was originally build with the idea to provide reliable and consistent global routing information, it is considered to be obsolete [12] to some point. This is because the IRR contains **static information** that is primarily fed by administrators from different ISP on a voluntary basis. Parts of the IRR are outdated, incorrect or simply missing. On the other hand, the IRR contains a lot of unique scientific information that is valuable.

RIS Routing Information Service The Routing Information Service (RIS) is a **dynamic** database primary focused on storing routing data for troubleshooting and research. This includes solving maintenance issues like outages or problems like route flapping. RIS started at October 1999 and collects information by deployment of several Remote Route Collectors (RRC) at many Internet Exchanges 2. The RRC are passively monitoring many BGP peers and provide a unique view of the Internet, similar to Oregon Route Views.

Currently RIPE NCC has 15 remote route collectors (RRC) that peer with 600 collector peers (CP) which collect routing tables of their customers. Each RRC contains RIB information, BGP updates and BGP status changes. The purpose of RIS within the project of the INRB to monitor actual usage of ASN and IP addresses within a timeframe.

2 INRDB PROTOTYPE

The experimental Internet Number Resource database (INRDB) holds raw data about address space allocation, routing registry information and other related objects. The purpose of the INRDB is to provide as much information as possible about number resources from a single point, by combining data from the RIPE DB, RIS and various other sources.

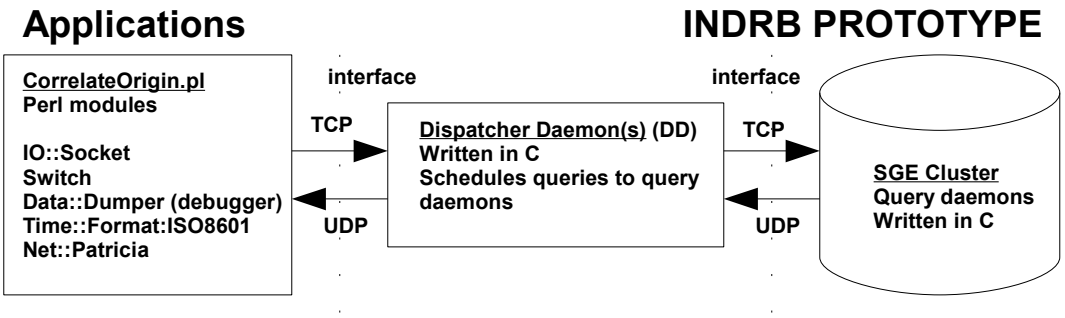


Figure 1: The INRDB can be queried using user applications. Queries are redirected by the dispatcher daemon (DD) to a query daemon (QD) in the cluster.

Other goals of the INRDB are:

- provide information about data sources in a minimum time
- establishing relations between data sources, measuring quality and resolve inconsistencies
- scalability, allow parallel processing of querying different data sources and open-end design
- improving the quality from underlying data sources like RIPE_DB and RIS

Some of the challenges are:

- a lot of overlapping information and different kinds of data
- enormous amount of data as input for INRDB, especially RIS data: in the amounts of terabytes

2.1 DATA MODEL AND SYSTEM ARCHITECTURE

The current INRDB design 2 has three different levels of data storage and retrieval.

Component 1 Number resource index; retrieve IP prefix (specific subtree) information
Component 2 Time index, stores time related information (like validity time) in SQL.
Component 3 Blob store(s), stores blob data of one particular data class (like RIPE_DB objects)

The RIS data for resource collectors contain billions of records and the amount of data that is used as input for the INRDB is enormous. That amount of data could not be processed in a acceptable time by only using SQL. Therefore, the RIPE NCC Science came up with a new solution by partly storing number resource information in memory. This involves the following phases:

- the update process retrieves information from various data sources (RIS, RIPE_DB, RIR_STATS) and stores them into more condensed structures. This involves three phases. The preprocessing phase is a manual task of rsyncing all data from various data sources for processing. The processing phase is the most time and memory consuming and is aggregated in parallel in a cluster.

The processing phase stores blobs, time intervals, IP and ASN information in SQL structures. The postprocessing phase converts the SQL data to highly optimized memory structures which are a lot faster to query. The idea is that blob indexes and mappings, IP trees and time intervals are stored in memory, while the actual data is stored on disk.

- the query process answers queries from users (figure 2) and presents answers based on previously processed information by connecting to a dispatcher daemon (DD). The requests itself are dispatched from the dispatcher daemon using a back-end channel to all query daemons (QD) it knows about.

DATA REPRESENTATION The basic unit of information is a blob (binary large object) which falls in a certain object data class. Each dataclass may have multiple data sources (RIS peer, ARIN, RIPE NCC etc.). Blobs are grouped in blob collections that are saved in compressed raw ASCII files. In this document we only describe the high level classes RIS_RIB, RIR_STATS and RIPE_DB. Data classes can be divided into individual types of data, like inetnum objects or route objects.

RIPEDB Data Objects	
as-block	blocks of AS numbers held by RIRs; basically IANA allocations but also (ERX) blocks.
aut-num	AS numbers added by RIPE NCC or users from other regions who use RIPE routing registry
inetnum	IP address blocks allocated/assigned by the RIPE NCC
route	specified by interAS routes contains prefix route and origin AS
domains	Reverse address lookup (x.y.z.in-addr.arpa)

Route object

```
BLOB (RIPE_DB): route: 193.0.0.0/xx
descr:    RIPE-NCC
origin:   AS3333
mnt-by:   RIPE-NCC-MNT
changed:  xxxxxxxxxxxx  19960812
changed:  xxxxxxxxxxxx  20000908
source:   RIPE
2001-09-22T09:33:24Z-2007-10-15T00:00:00Z
```

Inetnum object

```
inetnum:   193.0.0.0 - 193.0.1.255
netname:   RIPE-NCC
descr:     RIPE Network Coordination Centre
descr:     Amsterdam, Netherlands
country:   NL
admin-c:   DDL122-RIPE
tech-c:    OPS4-RIPE
status:    ASSIGNED PI
remarks:   until 19990305
mnt-by:    RIPE-NCC-MNT
mnt-lower: RIPE-NCC-MNT
changed:   xxxxxxxxxxxxxxxxx 19960815
changed:   xxxxxxxxxxxxxxxxx 20020410
source:    RIPE
2002-04-10T12:47:09Z-2003-03-17T11:57:06Z
```

RIS Data

```
BLOB (RIS_RIB, 202.249.2.xx@rrc06):
TABLE_DUMP2||B|202.249.2.xx|4777|
193.0.12.0/xx|4777 2497 3549 1103 3333|
2007-09-18T00:00:00Z - 2007-09-18T00:00:00Z
```

aut-num object

```
BLOB (RIPE_DB): aut-num: AS3333
as-name:  RIPE-NCC-AS
descr:    RIPE Network Coordination Centre
import:   from ASxxxxx
          195.69.144.2xx at 195.69.144.xx
          action pref=100;
          accept ANY
export:   to ASxxxxx
          195.69.144.2xx at 195.69.144.xx
          announce AS3333
mnt-by:   RIPE-NCC-MNT
changed:  xxxxxxxxxxxxxxxxx 20041214
source:   RIPE
2004-12-14T15:52:58Z-2004-12-28T17:19:31Z
```

RIS RRC Raw data captured with Quagga

```
TIME: 10/01/07 07:59:55
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 12.178.27.0/xx
SEQUENCE: 2092
FROM: 202.12.28.190 ASxxxxx
ORIGINATED: 09/25/07 10:46:22
ORIGIN: IGP
ASPATH: 2497 3356 14745 20141 20141 11065
NEXT_HOP: 202.12.28.xxx
```

RIR Statistics Exchange Format

```
registry|cc|type|address|length|date|status
BLOB (RIR_STATS):
ripe|EU|ipv4|193.0.0.x|256|19930901|assigned
2003-11-26T00:00:00Z-2007-08-24T00:00:00Z

BLOB (RIR_STATS):
ripe|EU|asn|33xx|1|19940519|allocated
2006-03-30T00:00:00Z-2007-08-24T00:00:00Z
2007-09-02T00:00:00Z-2007-09-05T00:00:00Z
```

Figure 2: Relevant RIPEDB, RIS, RIR_STATS objects, classes and properties

RIPE DB The RIPE_DB has 21 different types of data, but the INRDB contain only the ones relevant to number resources [2]. Each object type holds information about organizations that hold the resources, where the allocations were made and contact details.

Routing Information Base updates are collected and presented using the TABLE_DUMP_V2 Type from the MRT routing information export format [15]. For example: RIS Data in [2] gives us the

collector peer (CP) IP: 202.249.2.xx and it's ASN 4777. The traversed AS_PATH is 4777 2497 3549 1103 3333 and AS_ORIGIN 3333 which announced the route. The relation between peers is described as AS_LINKS.

RIR Statistics are collected and presented using the RIR Statistics Exchange Format. RIR statistics [2] give more reliable information about assigned prefixes or allocated ASN. The format describes a data source (IANA, APNIC, RIPENCC ...), two character country code, number resource, address, prefix length/number ASN in range and status.

2.2 INCONSISTENCIES

The definition of an inconsistency in this report refers to data that is **semantically** incorrect, inaccurate or different than the data found in other data sources. For the static RIPEDB, most intra-DB inconsistencies are caused by abandoned data, user errors and change of policy rules. Inconsistencies can be overlapping inetnum objects, unreferenced contact info or wrong notation of number resources.

Dynamic intra RIS inconsistencies are caused by other means because they reflect the routing topology of the real world. Intra RIS inconsistencies can be caused by millions of small timeframes that may overlap, regional topology changes and time delays. Other inconsistencies are different AS announcing the same prefix. It could also be an AS that announces a prefix that overlaps address space of other prefixes.

Until this point, we only discussed intra datasource inconsistencies. Comparing data sources brings in a whole new range of inconsistencies. For example, a prefix-AS tuple that is just registered will not be immediately be routed and listed in the RIPEDB, which takes time. Timeframes between data sources can also overlap, introduce holes or expire.

The author of this report has chosen to only look at specific cases: the number of MOAS in RIS, conflicting AS Origins and unregistered prefix usage. Complexity of timeframes is reduced by measuring in hourly sample times.

3 VARIATIONS AND PROPERTIES OF PREFIX HIJACKING

Article(s)[5][6][7] already point out the various combinations of possible hijacking scenarios. These can be summarized in hijacking only a prefix, hijacking the prefix and AS, hijacking the subnet of a particular prefix (and AS) or hijack a supernet of a particular prefix (with or without ASN). Captured ASN are considered undetectable on the control plane without any additional means like resource certificates. AS_ORIGINS (endnodes) which announce unauthorized prefixes will result in conflicting MOAS which can be detected. Unfortunately, there are also legitimate cases of MOAS[5].

For example, a customer with private ASN or static links, or single/multihomed aggregated prefixes. Note that this does not mean that multihoming always causes MOAS conflicts, this entirely depends of the configuration of the customer. Other legitimate MOAS cases can be caused by anycasting [8] which has gained increased popularity among DNS root servers. False positives due to anycasting can be easily avoided by filtering 'anycast addresses'. To narrow down our search we further define properties of hijacked prefixes:

- Hijacked prefixes are mostly dynamic and stealthy and do not correlate with history, while legitimate routes in the control plane are more stable for a long period. Hijacked prefixes have a short uptime.
- Hijacked prefixes are mostly small /24 from organizations that are not aware that the address space is used by someone else.
- Any detector that receives a prefix knows it's BGP neighbors so can detect false last hops. (dataplane)
- Many hijacked prefixes involve sub/supernet address space. Overlap detection can be done with **radix trees**. [2]
- RIS prefixes that are routed/used but unregistered and MOAS cases with multiple overlapping timeframes with origins not listed in RIPEDB are suspicious.

[2]Patricia Trie data structure to quickly perform IP address prefix matching for applications such as IP subnet, network or routing table lookups

4 METHODOLOGY

The following algorithm describes a way to retrieve the amount of unique RIS prefixes, unique RIS prefixes with unlisted RIPEDB origins and unique RIS prefixes that are used while not registered.[3]. The whole trick to use datastructures of hash key sequences to construct a hierarchical tree and is the central part of the algorithm to filter millions of RIS entries that may have identical or unique prefixes, single or multiple AS (MOAS) or multiple (non unique) timeframes. Note that this algorithm does not support overlapping subnet/supernet detection and does not compare time intervals. Results are plotted with the GD::Graph Perl module.

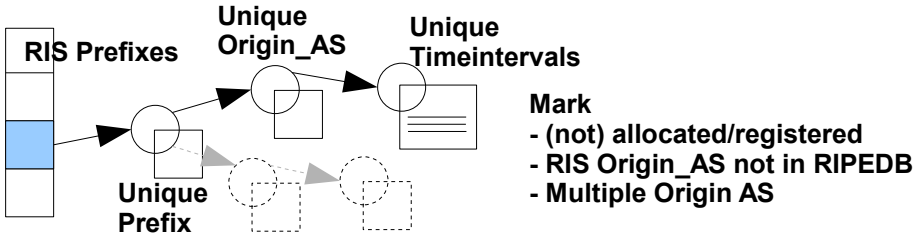


Figure 3: **The algorithm generates sequences of hashes using only hash keys to store information. The key values are used as keys. The unique hashes create new leaves and identical ones are overwritten.**

Algorithm 1: Retrieves percentage of unregistered prefix usage, prefixes with OriginAS not listed in RIPEDB and unique MOAS. Source code is listed in 5.1

```
Input: prefix list P, sampletime list T
Output: percentage of unregistered prefix usage, prefixes with OriginAS not listed in RIPEDB
        and unique MOAS
Query RIS subtree P on more specifics for sampletime T and store results in hash tree;
foreach unique RIS prefix Px in RIS subtree P do
    Query RIR_STATS for exact match or less specifics on T ;
    Query the RIPEDB for exact match or less specifics on T ;
    if Px is registered/allocated then
        foreach Origin_AS in unique RIS prefix Px do
            if RIS_ORIGIN_AS matches RIPEDB_ORIGIN_AS then
                Mark the RIS_ORIGIN_AS as listed in RIPEDB
            end
        end
    end
    else
        prefix is not registered/allocated, add to suspect list
    end
end
Search the constructed hash tree for MOAS within T and add those to the suspect list.;
```

[3]Hardware: Pentium Dual Core 1.8GHZ/2GB laptop; Running Ubuntu Feisty with kernel 2.6.22-14

5 RESULTS AND CONCLUSIONS

The plots [5] show a four year overview of historical RIS data of 62/8 and one hour samples of all /8 allocated to RIPE NCC. 62/8 was chosen because it was allocated to RIPE NCC in April 97. Early allocated address space have a higher probability rate to actually find results. There were also anomalies found like queries on route objects that returned 839 route blobs in the RIPEDB. Other problems were stability problems in INRDB itself and some gaps for RIS information. Some high ratings for unregistered prefix usage are not entirely accurate because some prefixes are listed in RIR_STATS on such a way that it's not detectable by the algorithm (for example lacking CIDR notation).

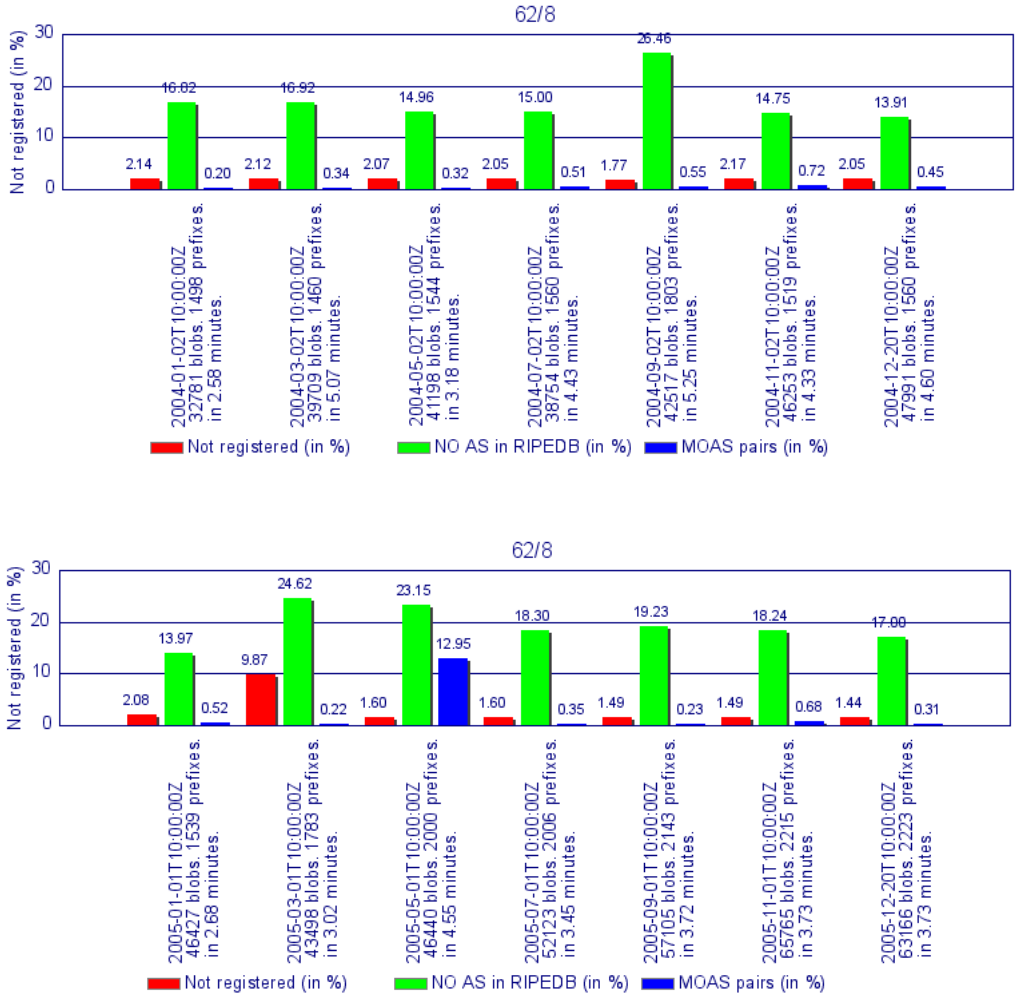


Figure 4: **Overview 62/8 2004-2005, shows a history of two years of samples with the number of unique prefixes in the subtree, the percentage that was used and not registered, the percentage of unique MOAS cases and the percentage of unique prefixes with no matching Origin in RIPEDB. The results are largely the same, except for one case that has 12% MOAS. Similar MOAS cases were found during the Google 2005 Outage [8]**

The plots show the RIS RIB table growth 62/8 from 1495 unique prefixes in 2004 to 2166 unique prefixes in 2007. The quality of the RIPEDB also seems better than expected. The average of 88% RIS entries that match AS_ORIGINS in RIPEDB is a positive result. Especially if you consider the amount of RIS bogus announcements.

Although measuring sample information provides us with lot new and unique information, there is also lot of information ignored. The author believes that measuring in timeframes, detecting overlaps and filtering sub/supernet overlapping gives a much more detailed and efficient results.

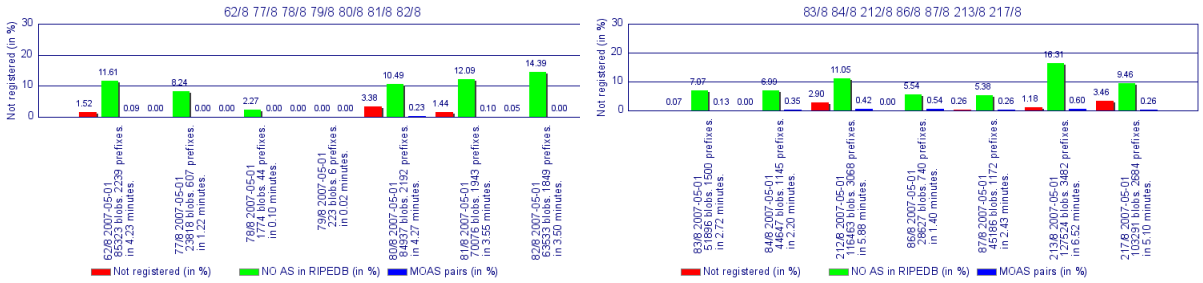


Figure 5: **Comparison between multiple /8 on 01-10-2006 with the number of unique prefixes in the subtree, the percentage that was used and not registered, the percentage of unique MOAS cases and the percentage of unique prefixes with no matching Origin in RIPEDB.**

Also, without resource certificate validation and analyzing repeating patterns, there can never be full reliable estimate of prefix hijacking. With other words, resource certification is a elementary point for proving illegitimate resource usage. Thus, we conclude and answer our research question[1] that there is currently not enough information to actually determine whether there is a rising trend in malicious activity.

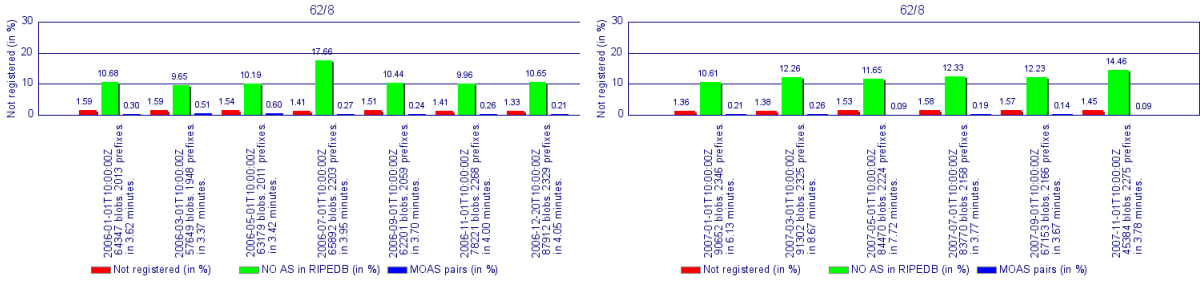


Figure 6: **Overview 62/8 2006-2007**

The MOAS results of the implementation of the algorithm (which are not listed here for ethical reasons) showed that almost 80% percent of all MOAS keeps coming back every month. However, correlating these MOAS cases with listings in RIPEDB and registration data is not enough to determine if they are hijacked or not. For this we need overlap detection, timeframe processing and resource certification validation.

5.1 FUTURE WORK

Future work is overlap detection, timeframe processing and resource certification validation. Part of future work that requires less time could be examining if the resulting 20% repeats MOAS behavior (repeatedly hijacking more prefixes) or to filter MOAS on bogons. The author of this document hopes that this short project is a small step in wide array of new research concerning the INRDB and that we can one day solve the problem of prefix hijacking if all organizations work together by deploying resource certification.

References

[1] *An Analysis of BGP Multiple Origin AS (MOAS) Conflicts*
Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, Lixia Zhang, 2001
http://www.imconf.net/imw-2001/imw2001-papers/88.pdf

[2] *Analyzing BGP Policies: Methodology and Tool*
Proceedings of IEEE INFOCOM, Hong Kong, China, March 2004.
http://www.cs.ucr.edu/~siganos/papers/Nemecis.pdf

[3] *A Blueprint for Improving the Robustness of Internet Routing*
Georgos Siganos, Michalis Faloutsos , 2005
http://www.cs.ucr.edu/~siganos/papers/security06.pdf

[4] *A Distributed Reputation Approach to Cooperative Internet Routing Protection*
Harlan Yu Jennifer Rexford Edward W. Felten, Princeton University , 2005
http://www.cs.princeton.edu/~jrex/papers/npsec05.pdf

[5] *Accurate real-time identification of IP prefix hijacking*
Xin Hu, Z. Morley Mao, University of Michigan
http://www.eecs.umich.edu/techreports/cse/2006/CSE-TR-516-06.pdf

[6] *A study of prefix hijacking an Interception in the Internet*
Hitesh Ballani, Paul Francis, Xinyang Zhang, Cornell University, 2007
http://www.cs.cornell.edu/People/francis/sigcomm07-interception.pdf

[7] *A lightweight distributed scheme for detecting IP prefix hijacking in real-time*
Changxi Zhen, Lusheng Ji, Dan Pei, Jia Wang, Paul Francis, 2007
http://www.sigcomm.org/ccr/drupal/files/fp324-zheng.pdf

[8] *Analysis of BGP Prefix Origins During Googles May 2005 Outage*
Tao Wan Paul, C. van Oorschot, Carleton University, 2005
http://www.scs.carleton.ca/~paulv/papers/ssn06-fine.pdf

[9] *Beware of BGP Attacks)*
Ola Nordstrom, Constantinos Dovrolis, College of Computing
http://www.cc.gatech.edu/~dovrolis/Papers/ccr-bgp.pdf

[10] *Detecting Bogus BGP Route Information: Going Beyond Prefix Hijacking*
Jian Qiu, Lixin Gao *et al*, Department of ECE, Univ. of Massachusetts, 2007
http://www.ece.rice.edu/~sranjan/publications/securecomm07-hijacking.pdf

[11] *How prevalent is prefix hijacking on the internet?*
Peter Boothe, James Hiebert, Randy Bush
http://rip.psg.com/~randy/030603.nanog-sxbgp.pdf

[12] *How to extract BGP peering information from the internet routing registry*
Giuseppe Di Battista, Tiziana Refice and Massimo Rimondini, University of Roma Tre
http://portal.acm.org/ft_gateway.cfm?id=1162685&type=pdf&coll=portal&dl=ACM&CFID=15151515&CFTOKEN=6184618

[13] *Internet Routing Security Issues and Requirements Definition*
Feki, Achemlal France telecom Research, 2006
http://www.temu.gr/2006/sessions%5C2%5C2%20ID%201502.pdf

[14] *Learning Perl, 4th Edition*
O'Reilly, July 2005
http://www.unix.org.ua/orelly/perl/learn/index.htm

[15] *MRT routing information export format*
IETF Network group
http://tools.ietf.org/id/draft-ietf-grow-mrt-04.txt

[16] *PHAS: A Prefix Hijack Alert System*
M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, in Proceedings of 15th USENIX Security Symposium, 2006
http://irl.cs.ucla.edu/papers/originChange.pdf

[17] *Pretty Secure BGP (psBGP)*
Tao Wan Evangelos Kranakis, P.C. van Oorschot, Carleton University, Ottawa 2004
http://www.isoc.org/isoc/conferences/ndss/05/proceedings/papers/tao-psBGP.pdf

[18] *Pretty Good BGP: Improving BGP by Cautiously Adopting Routes*
Josh Karlin, Stephanie Forrest *et all*, University of New Mexico,Princeton, 2006
http://www.cs.princeton.edu/~jrex/papers/pgbgp.pdf

[19] *Routing Policy Specification Language (RPSL)*
RFC 2622, Network Working Group
ftp://ftp.ripe.net/rfc/rfc2622.txt

[20] *RFC 4271 BGP4*
The Network Working Group, Januari 2006
http://www.rfc-editor.org/rfc/rfc4271.txt

[21] *RIPE NCC Science Group*
http://www.ripe.net/info/ncc/staff/science_grp.html

[22] *Teach Yourself Perl 5 in 21 days*
David Till, 2005
http://docs.rinet.ru/P7/

[23] *Using Resource Certificates*
Progress Report, Geoff Huston APNIC, October 2006
http://www.ripe.net/info/ncc/staff/science_grp.html

[24] *Securing BGP through Secure Origin BGP (soBGP)*
Cisco Internet Protocol Journal, September. 2003
ftp://ftp-eng.cisco.com/sobgp/presentations/BCR-soBGP.pdf

[25] *SBGP / SoBGP: What do we Really Need and how do we Architect a Compromise to get it*
Randy Bush, David Meyer et all, Nanog Salt Lake City, 2003

[26] *Understanding Resiliency of Internet Topology Against Prefix Hijack Attacks*
Mohit Lad y Ricardo Oliveira, Beichuan Zhang z Lixia Zhang, UCLA, 2006
http://irl.cs.ucla.edu/papers/hijack-dsn.pdf

```
#!/usr/bin/perl -w -I /home/peter/workspace/prefixCheck
# rir stats may give no feedback: see 192/20
# Patricia tree for overlapping
# parallel processing > /8
# uniform format?
# only ripe DB /8!!! not IANA, check IANA
# time: ISO8601 format YYYY-MM-DDTHH:mm:ssZ

package INRDB::App::PrefixCheck;
use warnings;
use diagnostics;
use strict;
use IO::Socket;
use base qw(Exporter);
use Switch;
use Data::Dumper;     # debug class
use DateTime::Format::ISO8601;
use GD::Graph::bars;
use GD::Graph::hbars;
use GD::Graph::Data;
use Data::Types qw(:all);

my $dispatcherHost = 'localhost';
my $dispatcherPort = '5556';
print "Executing $0 Please wait..\n";

my @xdata;
my ( @y1data, @y2data, @y3data, @y4data );

# IANA Allocation
# 62/8 Apr 97    RIPE NCC
my @prefixes = ( '193/8' );

my $sock;
my %uniMOAS;

# Sample times
my @sampletimes = ( "2007-05-01" );

foreach my $prefix (@prefixes) {
 our $prefix = $prefix;
 foreach my $sampledate (@sampletimes) {
  fetchINRDBdata( $sampledate, $prefix );
 }
}

sub fetchINRDBdata {

 # Open new connection
 $sock = query( $dispatcherHost, $dispatcherPort, '-k' );
 my $start = time;
 my %prefixes;
 my $risBlobs           = 0;
 my $risPrefixes        = 0;
 my $risPrefixesToCheck = 0;

 my @risPrefixesRegisteredAllocated;
 my @risPrefixesMatchOrigin;
 my @risPrefixesListedRIPEDB;
 my @risPrefixesMOAS;

 my ( $date, $inputPrefix ) = @_;
 DateTime::Format::ISO8601->parse_datetime($date);
 my $outputOptions = "+M +oc";
 my $timeOptions   = "+xT $date";
 my @ris_blobs     = query2( $inputPrefix, 'RIS_RIB', $outputOptions, $timeOptions, '-M' );

 foreach my $field (@ris_blobs) {
  if ( $field =~ /BLOB:/ ) {
   my ( $values, @timeintervals ) = split( 'VALID: ', $field );
   my ( $n1, $n2, $n3, $n4, $n5, $prefix_ris, $aspath_ris ) = split( '\|', $values );
   my @tmp = split( ' ', $aspath_ris );
   my $asorigin_ris = pop(@tmp);
   $prefixes{$prefix_ris}{$asorigin_ris} = ();
  }
 }

 $risBlobs    = @ris_blobs;
 $risPrefixes = keys %prefixes;

 # Generate false MOAS to test
 #$prefixes{'193.0.232.0/23'}{'6666666666666666'} = ();

 foreach my $prefix ( keys %prefixes ) {
  foreach my $origins ( keys %{ $prefixes{$prefix} } ) {
   if ( scalar( keys %{ $prefixes{$prefix} } ) > 1 ) {

    #print "prefix:", $prefix, ". \n";
    #print "MOAS: ", scalar( keys %{ $prefixes{$prefix} } ), "\n";
    my %seen;     # lookup table
    @seen{@risPrefixesMOAS} = ();
    foreach my $item (@risPrefixesMOAS) {
     $seen{$item} = 1;
    }
    if ( !exists $seen{$prefix} ) {
     push( @risPrefixesMOAS, $prefix );
     $uniMOAS{$prefix} = $date;
    }
   }
   $risPrefixesToCheck++;
  }
 }
```

```
103
104  print "Hold your breath there are: ", $risPrefixesToCheck . " to check..\n";
105  foreach my $prefix_ris ( keys %prefixes ) {
106   foreach my $asorigin_ris ( keys %{ $prefixes{$prefix_ris} } ) {
107    my @rir_blobs = query2( $prefix_ris, 'RIR_STATS', '', $outputOptions, $timeOptions, '' );
108    my @timeIntervals;
109    my $counter = -1;
110    foreach my $field (@rir_blobs) {
111     if ( $field =~ /BLOB:/ ) {
112      $counter++;
113      my ( $none, $none1, $none2, $address, $none3, $none4, $status ) = split( '\|', $field );
114      print "ass/all: ", $status, "\n";
115      if ( $status =~ 'assigned' || $status =~ 'allocated' ) {
116       print "address registration: ", $address, " ", $status, "\n";
117       $prefixes{$prefix_ris}{'registered'}{$address} = ();
118      }
119     }
120    }
121    print "Matched ", scalar(@rir_blobs), " RIR blobs\n";
122
123    # Query for exact match and automatically for 1-layer less specific
124    my @ripedb_blobs = query2( $prefix_ris, 'RIPE_DB +ds route', $outputOptions, $timeOptions, '' );
125    if ( !defined( $ripedb_blobs[0] ) ) {
126     print "Not listed in RIPEDB\n:";
127    }
128    else {
129     push( @risPrefixesListedRIPEDB, $inputPrefix );
130     $counter = 0;
131     foreach my $field (@ripedb_blobs) {
132
133      #print "FIELD: ",$field,"\n";
134      if ( $field =~ /BLOB:/ ) {
135       if ( $field =~ /route:.*\n/ ) {
136        my $route = ( split( ':\s', $& ) )[1];
137        $route =~ s/\s+//;
138        $route =~ s/\|//;
139        $route =~ s/\n//;
140        print $route, "\n";
141       }
142      }
143      if ( $field =~ /origin:.*\n/ ) {
144       my $origin = ( split( ':\s', $& ) )[1];
145       $origin       =~ s/\s+//;
146       $origin       =~ s/\|//;
147       $origin       =~ s/AS//;
148       $origin       =~ s/\n//;
149       $asorigin_ris =~ s/\{//;
150       $asorigin_ris =~ s/\}//;
151       if ( $origin =~ $asorigin_ris ) {
152        $prefixes{$prefix_ris}{$asorigin_ris}{'origin_match'} = $origin;
153       }
154
155      }
156      if ( $field =~ /VALID:.*\n/ ) {
157       my $tmp = ( split( ':\s', $& ) )[1];
158       $tmp =~ s/\|//;
159       $tmp =~ s/\n//;
160
161       #$ripedb_blobs[$counter]{valid} = $tmp;
162       $counter++;
163      }
164     }
165     print "Matching ", scalar(@ripedb_blobs), " RIPEDB route blobs\n";
166    }
167    $risPrefixesToCheck--;
168    print "Prefixes to query..", $risPrefixesToCheck, "\n\n\n\n";
169   }
170  }
171  foreach my $prefix ( keys %prefixes ) {
172   foreach my $origins ( keys %{ $prefixes{$prefix} } ) {
173    if ( $origins =~ 'registered' ) {
174     push( @risPrefixesRegisteredAllocated, $prefix );
175    }
176    foreach my $status ( keys %{ $prefixes{$prefix}{$origins} } ) {
177     if ( $status =~ 'origin_match' ) {
178      push( @risPrefixesMatchOrigin, $prefix );
179 # Store the number of prefixes that matches RIPEDB origin
180     }
181    }
182   }
183  }
184
185  if ( scalar(@risPrefixesRegisteredAllocated) != 0 ) {
186   my $percentall = 100 - ( ( scalar(@risPrefixesRegisteredAllocated) / $risPrefixes ) * 100 );
187   push( @y1data, to_float( $percentall, 2 ) );
188   print "allocated or registered: ", scalar(@risPrefixesRegisteredAllocated), "
189 prefixes: ", $risPrefixes, "\n";
190  }
191  else {
192   push( @y1data, to_float( 0, 2 ) );
193  }
194  if ( scalar(@risPrefixesMatchOrigin) != 0 ) {
195   my $percentNoOr = 100 - ( ( scalar(@risPrefixesMatchOrigin) / $risPrefixes ) * 100 );
196   push( @y2data, to_float( $percentNoOr, 2 ) );
197   print "match org: ", scalar(@risPrefixesMatchOrigin), " prefixes: ", $risPrefixes, "\n";
198  }
199  else {
200   push( @y2data, to_float( 0, 2 ) );
201  }
202  if ( scalar(@risPrefixesMOAS) != 0 ) {
203   my $percentMOAS = ( ( scalar(@risPrefixesMOAS) / $risPrefixes ) * 100 );
204   push( @y3data, to_float( $percentMOAS, 2 ) );
205   print "moas ", scalar(@risPrefixesMOAS), " prefixes: ", $risPrefixes, "\n";
```

```
206   }
207   else {
208    push( @y3data, to_float( 0, 2 ) );
209   }
210
211   #print Dumper(\%prefixes);
212   my $elapsed = to_float( ( time - $start ) / 60, 2 );
213   push( @xdata, "$date \n $risBlobs blobs. $risPrefixes prefixes. \n in $elapsed minutes.", );
214   close($sock);
215
216   graph2( [@xdata], [@y1data], [@y2data], [@y3data], "", "Not registered (in %)",
217  "NO AS in RIPEDB (in %)",, "MOAS pairs (in %)", "$inputPrefix", 'legenda' );
218  }
219
220  sub query2 {
221   my ( $prefix, $dataClass, $outputOptions, $timeOptions, $resourceOptions ) = @_;
222   my @blobs;
223   my $query = "+dc $dataClass $outputOptions $timeOptions $resourceOptions $prefix";
224   print $query, "\n";
225   print $sock $query, "\n";
226   while (<$sock>) {
227    s/\007//\n/g;
228    if ( $_ =~ /SUM/ ) {
229     last;
230    }
231    if ( $_ =~ /BACKEND !(OK|NQ)/ ) {
232     print "error\n for $prefix";
233     exit;
234    }
235    push( @blobs, $_ );
236   }
237   if ( defined( $blobs[0] ) ) {
238    return @blobs;
239   }
240   else {
241    return 0;
242   }
243  }
244
245  sub query {
246   my ( $ddhost, $ddport, $cmd ) = @_;
247   my $sock = new IO::Socket::INET(
248    PeerAddr => $ddhost,
249    PeerPort => $ddport,
250    Proto    => 'tcp',
251    )
252    or die "Cannot connect to DD";
253   print $sock $cmd, "\n";
254   return $sock;
255  }
256
257  sub graph2 {
258   my ( $xdata, $y1data, $y2data, $y3data, $xlabel, $y1label, $y2label, $y3label, $title, $legenda ) = @_;
259   my $data = GD::Graph::Data->new( [ [ @{$xdata} ], [ @{$y1data} ], [ @{$y2data} ], [ @{$y3data} ] ] )
260     or die GD::Graph::Data->error;
261   my $values    = $data->copy();
262   my $my_graph = GD::Graph::bars->new( 800, 400 );
263   my $name      = "193slash8 20070501";
264   $name =~ s/\./-/g;
265   $name =~ s/\//slash/g;
266   $name =~ s/\s/-/g;
267
268   #print STDERR "Processing $name\n";
269   $my_graph->set(
270    x_label  => $xlabel,
271    y1_label => $y1label,
272    y2_label => $y2label,
273
274    #y1_max_value      => 30,
275    #y2_max_value      => 30,
276    y_max_value   => 30,
277    title         => $title,
278    y_tick_number => 8,
279
280    #long_ticks        => 1,
281    #y_label_skip      => 2,
282    bar_spacing       => 6,
283    shadow_depth      => 2,
284    bargroup_spacing => 6,
285    accent_treshold  => 200,
286
287    #x_label_position => 1 / 2,
288    transparent => 0,
289    l_margin     => 20,
290    b_margin     => 20,
291    r_margin     => 20,
292    t_margin     => 40,
293    show_values => 1,
294
295    #two_axes => 1,
296
297    y_tick_number => 3,
298    y_label_skip  => 1,
299    x_plot_values => 1,
300    y_plot_values => 1,
301
302    long_ticks         => 1,
303    x_ticks            => 0,
304    x_labels_vertical => 1,
305
306    legend_marker_width => 24,
307    line_width          => 3,
308    marker_size         => 5,
```

```
309
310     #legend_placement => 'RC',
311   ) or warn $my_graph->error;
312
313   $my_graph->set_legend( $y1label, $y2label, $y3label );
314   my $font_spec = '/usr/share/fonts/truetype/msttcorefonts/arial.ttf';
315   $my_graph->set_y_label_font( $font_spec, 10 );
316   $my_graph->set_x_label_font( $font_spec, 10 );
317   $my_graph->set_y_axis_font( $font_spec, 10 );
318   $my_graph->set_x_axis_font( $font_spec, 10 );
319   $my_graph->set_title_font( $font_spec, 12 );
320   $my_graph->set_legend_font( $font_spec, 10 );
321   $my_graph->set_values_font( $font_spec, 8 );
322   $my_graph->plot($data) or die $my_graph->error;
323
324   local (*OUT);
325   my $ext = $my_graph->export_format;
326   open( OUT, ">$name.$ext" )
327     or die "Cannot open $name.$ext for write: $!";
328   binmode OUT;
329   print OUT $my_graph->gd->$ext();
330   close OUT;
331 }
332
333 #close($sock);
334
335 #foreach my $moas (keys %uniMOAS){
336 #       print "MOAS: ",$moas,": ",$uniMOAS{$moas},"\n";
337 #}
338 print "Done..\n";
```