Working Environment

Securing the Border Gateway Protocol BGPv4

Máster Interuniversitario de Seguridad de las TIC

Advisor: Joan Borrell Viader

Student: José María Foces Vivancos

June 2018

Contents

1. Introduction

This document contains the results and the steps followed in order to deploy a simulation composed by 22 routers, 8 network clients and a RPKI-RTR Server, deployed and sharing routes through BGPv4 without any security measures but with some prefix filters to implement iBGP route exchanges.

The purpose is to have a sandbox that can be easily changed and regenerated to evaluate the security impact of security measures. For example, TCP-MD5 or GTSM. In addition, we want it to provide an interface where changes on routing tables or traffic flows can be observed in real time and stored for further analyses.

1.1. Previous Work

Our first PoC resulted in four routers of four ASes (1,2,3 and 4) and used FRrouting BGP available implementation. It shown that it was enough to deploy routers with 128MB of RAM and that the hypervisor, Linux KVM, worked fine when deploying several switches to build complex networks.

1.2. Objectives

1. Define the simulated environment.
    a) The environment must simulate a portion of the internet with:
        i. A backbone connecting Tier 1 networks that peer free between each other.
        ii. Tier 2 network composed of 4 multi-homed ISPs.
        iii. Tier 3 network composed by single homed clients.
    b) The environment should not have security measures.
    c) Each machine deployed should have at least one way to accept incoming TCP connections and UDP datagrams.
    d) The routers must support RTR-RPKI Prefix Origin Validation.
2. Take advantage of the automation implemented previously and make improvements in order to deploy a more complex scenario.
3. Provide a way to observe flows over the whole systems and both kernel routing and BGP daemon route tables statues.
4. Build the simple simulation.

2. Work-Flow

2.1. Environment Definition.

The testing environment is composed of 29 machines. All of them can be accessed through the management network 172.16.1.0/24.

Addresses are outlined on the annex A.

The environment is composed of the following groups of machines:

2.1.1. Tier 1

a) 4 Internet exchange points. Numbered as 1,2,3 and 4. Identified by IX_N.

b) 4 Autonomous Systems (AKA ASes): 1,2,3 and 4 identified by ASN.

   i. Respectively they own <ASN>.0.0.0/8

   ii. They have one connection with each IX network.

   iii. They have two networks inside: Control and Downstream.

      1. Control is used to exchange routes. This maps directly to the Hypervisor network identified by control_<ASN>.

      2. Downstream is used to provide service to customers. This maps directly to a Hypervisor network identified by down_<ASN>.

   iv. Each AS has 4 routers: 1,2,3 and 4 identified by ASRouterCardinal. Each router:

      1. Is connected on Hypervisor networks:

         a) IX_<RouterCardinal>: to exchange routes with other Tier 1 ASes. They advertise <ASN>.0.0.0/8 network, not leaking internal structure. The idea is to have all routers with the same ASRouterCardinal connected on the same IX bridge. The interface name in the router is the same as on the Hypervisor.

         b) control_<ASN>: to exchange routes with routers that belong to the same ASN, referenced in this document as brothers. They advertise the full routing table but subnets. Customer networks won't be shared. All routers of the same AS are connected on the same control bridge. The name of this interface inside each router is control.

         c) down_<ASN>: To provide service to customers. They advertise and accept advertisements to anywhere. Each router has two NICs attached to each down_<ASN> bridge. The name of this interface inside each router is down and down_control. Down is used as downstream and down_control is used to perform the route exchange and Intra-AS traffic forwarding.

2. Owns the subnet <ASN>.((<ASRouterCardinal>-1)*64).0.0/10.

3. Routers with cardinals 2 and 4 delegate the subnet <ASN>.(((<ASRouterCardinal>-1)*64)+32).0.0/11 to a Tier 2 AS.

   a) They are upstream for multi-homed Tier2 ASes and therefore, they advertise all prefixes owned by the given Tier 2 AS.

2.1.2. Tier 2

a) 4 ASes: 21,22,23 and 24.

b) Each one has just 1 router, connected with 2 upstream tier 1 networks.

   i. These routers are connected on the following Hypervisor networks as follows:

      1. For each upstream they are connected on 2 down_<ASN> where ASN is the upstream. Inside these routers, these interfaces are called upstream_<ASN> where ASN is the upstream.

      2. They are also connected on another down_<ASN> bridge where ASN is its AS number. In this case they have two NICs connected on this network. Each NIC has assigned the IP address where they provide service to Tier 3 networks. Internally, the name of this interface is down.

c) They advertise both prefixes to both upstream routers.

d) The subnets assigned to each of them are described at Annex A.

2.1.3. Tier 3

a) Is composed by 8 single homed clients connected to one Tier 2 subnet.

b) They are dummy machines with static routing table to the upstream.

c) They are connected on downstream networks of tier 2 AS routers.

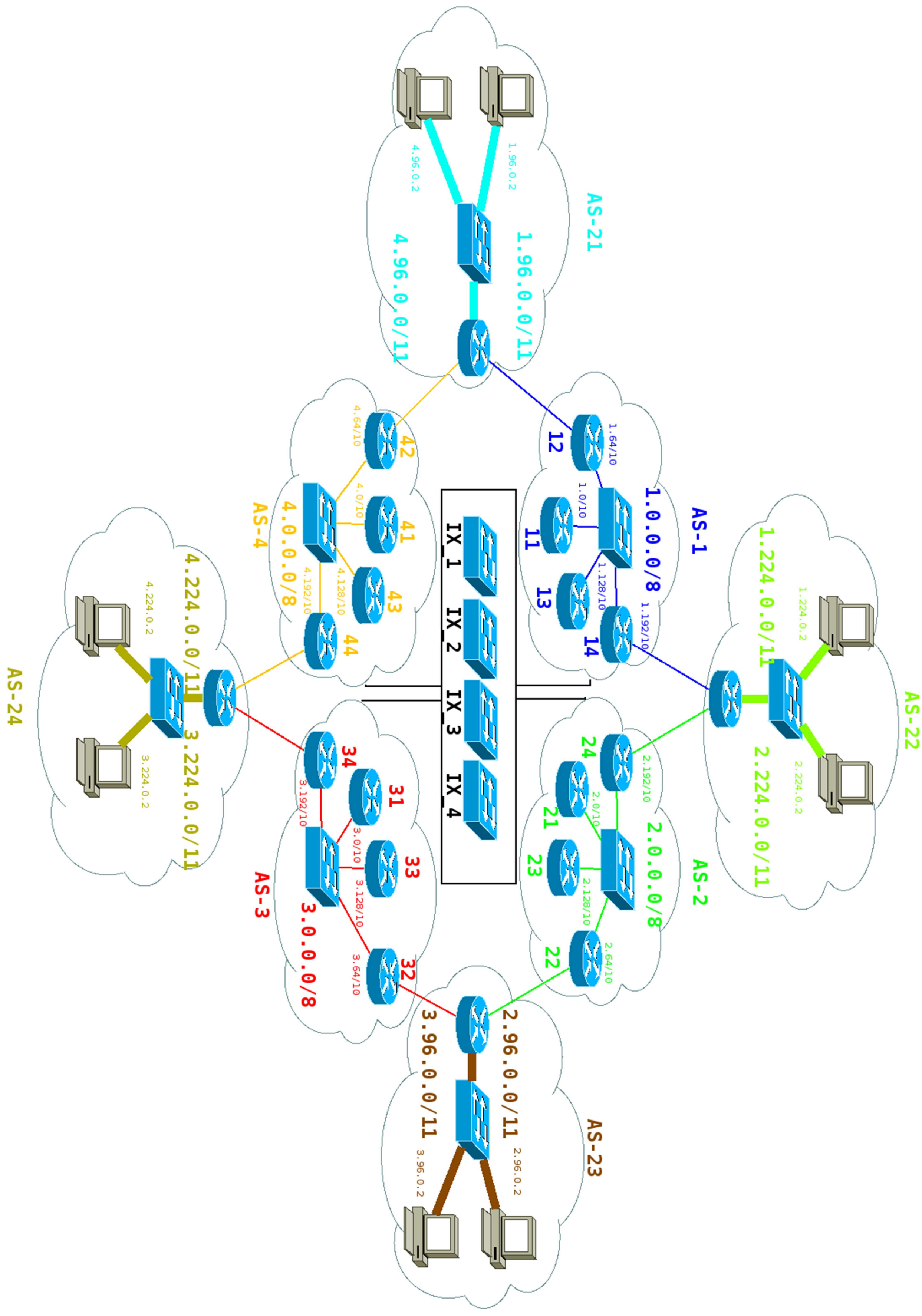
2.1.4. RPKI – RTR Server

a) To make things simpler, the server runs on the management network.

b) It's now shown on the diagram. Assume that it has connectivity with all machines through the management interface.

c) RPKI/RTR transfers are not done in a secure way through SSH Transport Security. This is not deployed in the environment .

The following diagram shows the topology:



2.2. ASes Router configuration examples

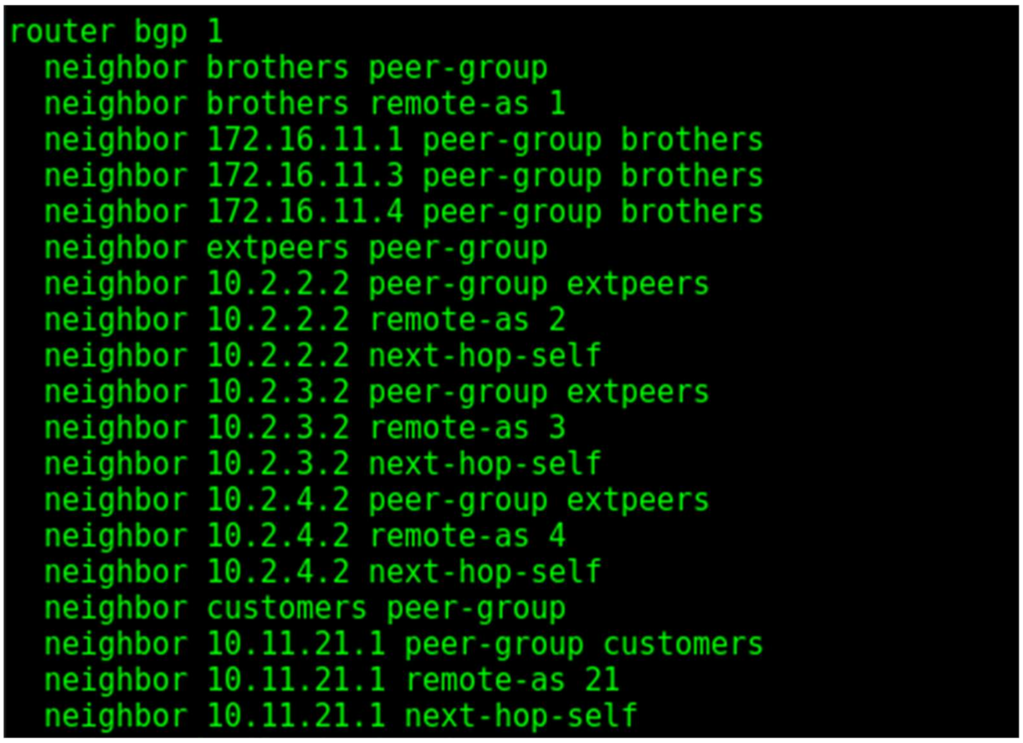
The automated deployment generates BGPd configuration files for each router. This configuration file is divided in the following sections:

2.2.1. Neighbors declaration

Grouping neighbors simplifies further configuration. In addition, next-hop-self indicates that the next-hop on the route should be replaced with the address of the router configured by this file. Otherwise, the hop willing to send traffic to the routes advertised from this one won't be able to communicate with it, since they are not in the same network segment.

This section just configures neighbors and neighbor groups.

Each neighbor declared here has to be configured with the remote AS number it belongs to.


```
router bgp 1
  neighbor brothers peer-group
  neighbor brothers remote-as 1
  neighbor 172.16.11.1 peer-group brothers
  neighbor 172.16.11.3 peer-group brothers
  neighbor 172.16.11.4 peer-group brothers
  neighbor extpeers peer-group
  neighbor 10.2.2.2 peer-group extpeers
  neighbor 10.2.2.2 remote-as 2
  neighbor 10.2.2.2 next-hop-self
  neighbor 10.2.3.2 peer-group extpeers
  neighbor 10.2.3.2 remote-as 3
  neighbor 10.2.3.2 next-hop-self
  neighbor 10.2.4.2 peer-group extpeers
  neighbor 10.2.4.2 remote-as 4
  neighbor 10.2.4.2 next-hop-self
  neighbor customers peer-group
  neighbor 10.11.21.1 peer-group customers
  neighbor 10.11.21.1 remote-as 21
  neighbor 10.11.21.1 next-hop-self
```

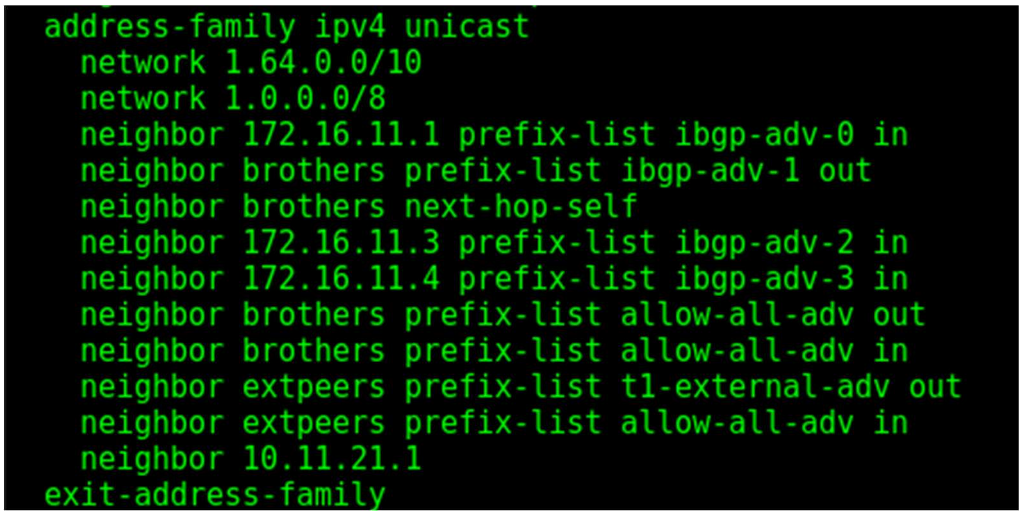
2.2.2. Setting network advertisements for IPv4 Unicast:

This section states the network advertisements to be announced by this router. It takes advantage of the grouping performed before.

In addition, it configures what prefixes are allowed to be advertised or accepted by this router.

Network configures the advertisement for the given prefix even if it doesn't know how to reach it.

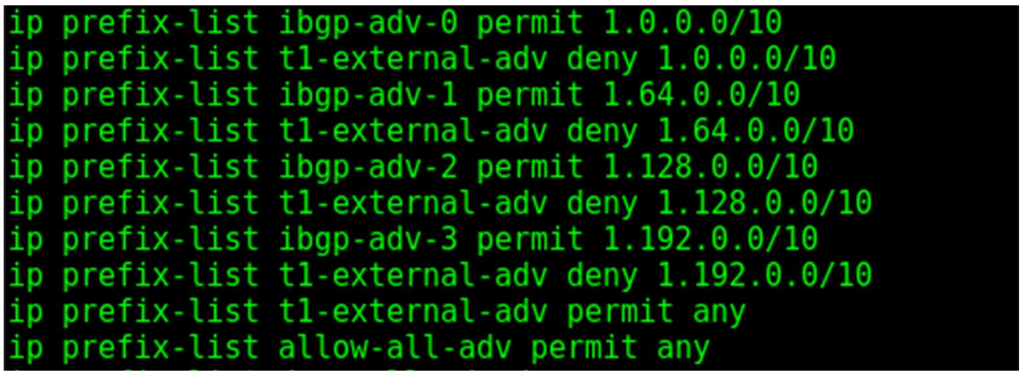
Neighbor configures the peering with the given router or group.



```
 address-family ipv4 unicast
  network 1.64.0.0/10
  network 1.0.0.0/8
  neighbor 172.16.11.1 prefix-list ibgp-adv-0 in
  neighbor brothers prefix-list ibgp-adv-1 out
  neighbor brothers next-hop-self
  neighbor 172.16.11.3 prefix-list ibgp-adv-2 in
  neighbor 172.16.11.4 prefix-list ibgp-adv-3 in
  neighbor brothers prefix-list allow-all-adv out
  neighbor brothers prefix-list allow-all-adv in
  neighbor extpeers prefix-list t1-external-adv out
  neighbor extpeers prefix-list allow-all-adv in
  neighbor 10.11.21.1
 exit-address-family
```

2.2.3. Prefix-lists declaration

Prefix-lists are configured to accept advertisements of internal AS1 network and to forbid the announcement outside AS1 (ibgp-adv-N). In addition, t1-external-adv defines that it does not accept routes to prefixes defined at ibgp-adv-N and to accept them otherwise. In addition, it trusts any advertisement made by AS21 (router 10.11.21.1)



```
ip prefix-list ibgp-adv-0 permit 1.0.0.0/10
ip prefix-list t1-external-adv deny 1.0.0.0/10
ip prefix-list ibgp-adv-1 permit 1.64.0.0/10
ip prefix-list t1-external-adv deny 1.64.0.0/10
ip prefix-list ibgp-adv-2 permit 1.128.0.0/10
ip prefix-list t1-external-adv deny 1.128.0.0/10
ip prefix-list ibgp-adv-3 permit 1.192.0.0/10
ip prefix-list t1-external-adv deny 1.192.0.0/10
ip prefix-list t1-external-adv permit any
ip prefix-list allow-all-adv permit any
```

2.3. Software installed.

The hypervisor has been upgraded and now has an Elasticsearch database plus the front-end named Kibana. This provides the interface to watch the full network operation without having to go machine-by-machine evaluating the changes.

All routers and clients have been upgraded and have installed Iperf, Netcat, Nginx and a route table watcher, implemented by me. All of them have been integrated with Systemd service manager.

2.3.1. RTRLib and Frrouting

In order to support RPKI-RTR protocol and therefore be able to construct environments implementing this security protocol, it's needed to compile both RTRLib and Frrouting. The process has three steps:

1. Prepare the system with needed tools

2. Compile & build Debian packages for RTRLib
3. Compile & build Debian packages for Frrouting to link against RTRLib

```
#!/bin/bash
## Step 1: Prepare the host.
apt-get -y install libjson-c-dev libsystemd-dev libc-ares2 libc-ares-dev flex libssh-dev
libssh-4 libreadline-dev pkg-config debhelper devscripts libncurses5-dev texlive-latex-base
texlive-generic-recommended libpam0g-dev libpam-dev libcap-dev imagemagick groff libpcre3-dev
chrpath libsnmp-dev dh-systemd doxygen python-sphinx cmake make gcc g++ linux-headers-amd64

## Step 2 RTRLib
git clone https://github.com/rtrlib/rtrlib.git
cd rtrlib
cmake -D CMAKE_BUILD_TYPE=Release .
make -j8
CMAKE_INSTALL_PREFIX:PATH=/usr/
debuild -b -uc -us

dpkg -i ../librtr0_0.5.0-1_amd64.deb
dpkg -i ../rtr-tools_0.5.0-1_amd64.deb
dpkg -i ../librtr-doc_0.5.0-1_all.deb
dpkg -i ../librtr-dev_0.5.0-1_amd64.deb

## Step 3 Frrouting
git clone https://github.com/FRRouting/frr.git
git checkout stable/4.0
./configure --with-pkg-extra-version=-RPKI
make dist
mv debianpkg debian
make -f debian/rules backports

mkdir frrpkg
cd frrpkg
tar xf ../frr_4.0-RPKI.orig.tar.gz
cd frr-4.0-RPKI/
. /etc/os-release
tar xf ~/frr/frr_4.0-RPKI-1~debian9+1.debian.tar.xz
debuild --set-envvar=WANT_RPKI=1 -b -uc -us
```

After that, we will have both Debian packages ready to be installed on the routers. After installing them, it's only needed to edit /etc/frr/daemons.conf to load the module with RPKI. (NOTE: This is a critical point. Otherwise there's no RPKI support)

```
bgpd_options="--daemon -A 127.0.0.1 -M /usr/lib/frr/modules/bgpd_rpki.so -f
/etc/frr/bgpd.conf"
```

2.3.2. RPKI-RTR-Server

The package (RPKI-RTR-Server) has been installed by converting the CentOS rpm package. The installation is straight forward.

```
#!/bin/bash
apt-get install alien
wget https://ftp.ripe.net/tools/rpki/validator3/beta/centos7/repo/rpki-rtr-server-3.0-
260.noarch.rpm
sudo alien --scripts rpki-rtr-server-3.0-260.noarch.rpm
```

This software allows to be configured to pull validated Route Origin Attestation (ROA) from a local validation server and act as RPKI – RTR Cache server. To adapt it to our needs we have defined the following JSON file and changed the configuration file to make it pull ROAs from the locally installed Nginx server that hosts this file:

Contents of /etc/rpki-rtr-server/application.properties:

```
server.port=8081
rtr.server.port=8323
rpki.validator.validated.objects.uri=http://localhost/validated.json¡
rtr.client.refresh.interval=3600
rtr.client.retry.interval=600
rtr.client.expire.interval=7200
logging.level.net.ripe.rpki.rtr=INFO
logging.level.org.springframework.context.annotation=INFO
```

Contents of the /var/www/html/validation.json:

```
{
  "data" : {
    "ready" : true,
    "trustAnchors" : [ ],
    "roas" : [ {
      "asn" : "AS1",
      "prefix" : "1.0.0.0/8",
      "maxLength" : 8,
      "links" : {}
    }, {
      "asn" : "AS2",
      "prefix" : "2.0.0.0/8",
      "maxLength" : 8,
      "links" : {}
    }, {
      "asn" : "AS3",
      "prefix" : "3.0.0.0/8",
      "maxLength" : 8,
      "links" : {}
    }, {
      "asn" : "AS4",
      "prefix" : "4.0.0.0/8",
      "maxLength" : 8,
      "links" : {}
    },{
      "asn" : "AS21",
      "prefix" : "1.96.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS21",
      "prefix" : "4.96.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS23",
      "prefix" : "3.96.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS23",
      "prefix" : "2.96.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS22",
      "prefix" : "1.224.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS22",
      "prefix" : "2.224.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS24",
      "prefix" : "3.224.0.0/11",
      "maxLength" : 11,
      "links" : {}
    },{
      "asn" : "AS24",
      "prefix" : "4.224.0.0/11",
      "maxLength" : 11,
      "links" : {}
    }
     ],
    "routerCertificates" : [ ]
  }
}
```

This file defines Prefix advertisement authorizations for this testing environment.

2.3.3. Iperf, Netcat, Nginx

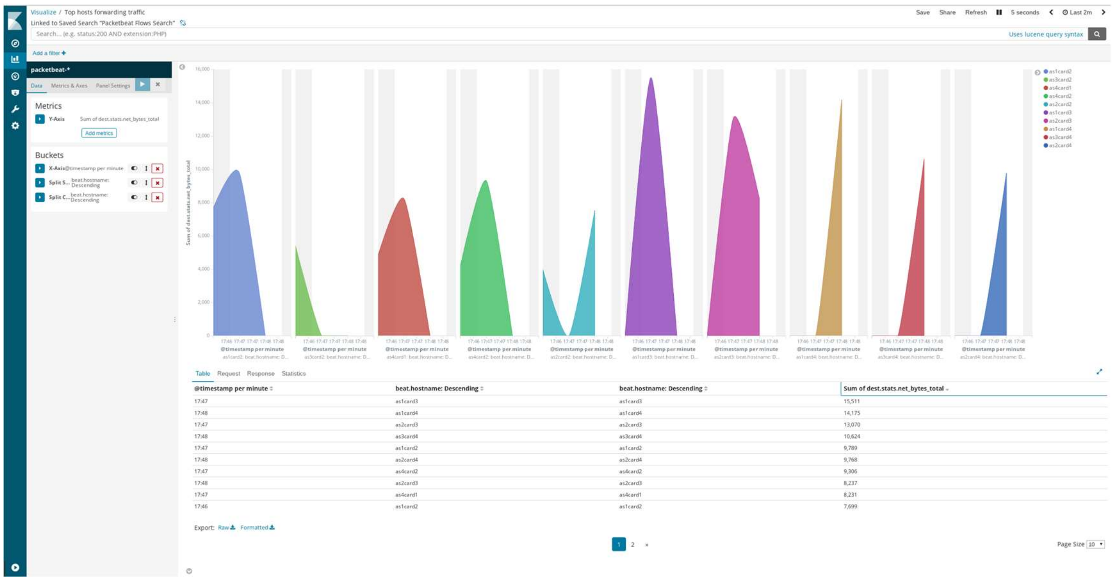
All of them are deployed as servers, so any machine in the network can send and receive traffic to and from them. They act as servers to be able to generate bi-directional traffic over the net.

In addition, Curl has been also installed to create HTTP requests against Nginx.

2.3.4. PacketBeat and Route Table Watcher

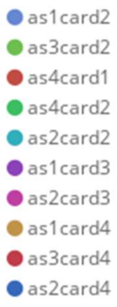
PacketBeat has been configured to send traffic metadata to the Elasticsearch database deployed on the hypervisor through the management network. This provides the following ways to see traffic flows information at high level over the whole network:

The image below provides an overview of a dashboard showing Top Traffic Forwarders, Receivers and Senders.


Visualize / Top hosts forwarding traffic
Linked to Saved Search "Packetbeat Flows Search"
Save
Share
Refresh
5 seconds
Last 2m
Add a filter +
packetbeat-*
Data
Metrics & Axes
Panel Settings
Metrics
Y-Axis
Sum of dest.stats.net_bytes_total
Add metrics
Buckets
X-Axis @timestamp per minute
Split S... beat.hostname: Descending
Split C... beat.hostname: Descending
@timestamp per minute
Table
Request
Response
Statistics

@timestamp per minute	beat.hostname: Descending	beat.hostname: Descending	Sum of dest.stats.net_bytes_total
17:47	as1card3	as1card3	15,511
17:48	as1card4	as1card4	14,175
17:47	as2card3	as2card3	13,070
17:48	as3card4	as3card4	10,624
17:47	as1card2	as1card2	9,789
17:48	as2card4	as2card4	9,768
17:47	as4card2	as4card2	9,306
17:48	as2card3	as2card3	8,237
17:47	as4card1	as4card1	8,231
17:46	as1card2	as1card2	7,699

Each bar shows the traffic volume over the top 10 traffic forwarders as follows:



● as1card2
● as3card2
● as4card1
● as4card2
● as2card2
● as1card3
● as2card3
● as1card4
● as3card4
● as2card4

Each bar shows the traffic volume over the top 10 traffic forwarders. On this example the following data was rendered:

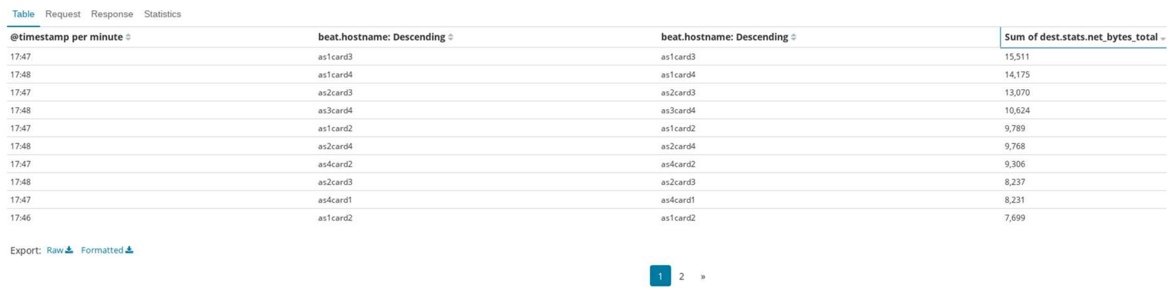


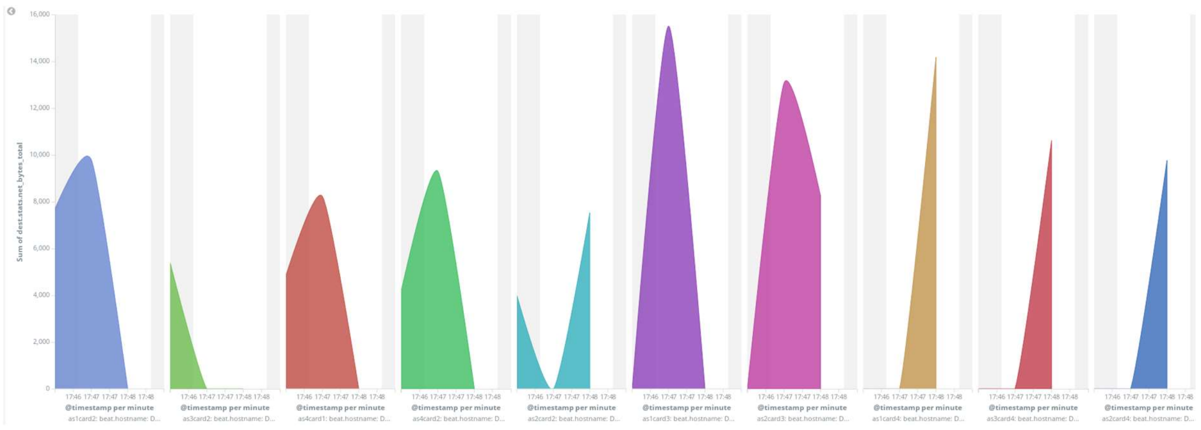
Table Request Response Statistics

@timestamp per minute	beat.hostname: Descending	beat.hostname: Descending	Sum of dest.stats.net_bytes_total
17:47	as1card3	as1card3	15,511
17:48	as1card4	as1card4	14,175
17:47	as2card3	as2card3	13,070
17:48	as3card4	as3card4	10,624
17:47	as1card2	as1card2	9,789
17:48	as2card4	as2card4	9,768
17:47	as4card2	as4card2	9,306
17:48	as2card3	as2card3	8,237
17:47	as4card1	as4card1	8,231
17:46	as1card2	as1card2	7,699

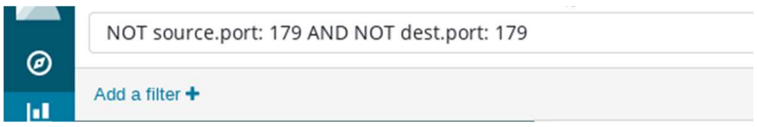
Export: Raw Formatted

1 2 »

The sums of forwarded bytes are rendered on the graphs as follows:



The full dashboard can be tuned to be updated on a certain interval, in real time and to show information from a relative and negative time offset from the current time. In addition, filters can be applied to match certain conditions over reported traffic by routers and client machines. For example, this matches any non-BGPv4 packets:

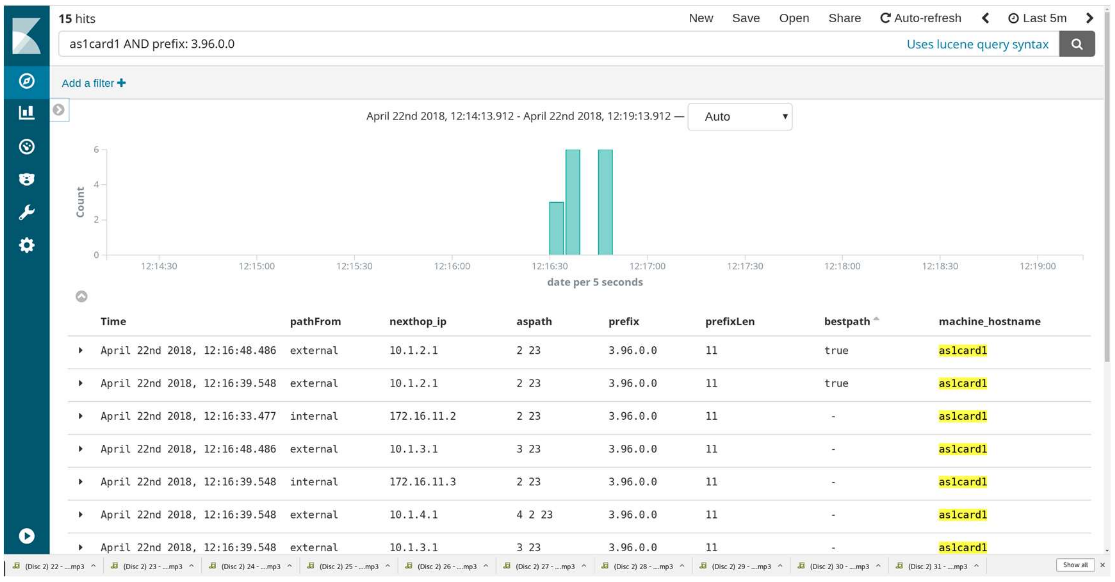


```
NOT source.port: 179 AND NOT dest.port: 179
```

Route Table Watcher offers similar capabilities than Packetbeat. Instead of sending beats about traffic, it sends beats about the status of both kernel routing table and BGP routing table handled by Frrouting: respectively using the commands ip route and vtysh show ip bgp json.

The software just updates any change when it takes place.

The following dashboard shows the routes received by as1card1 and a time-line:

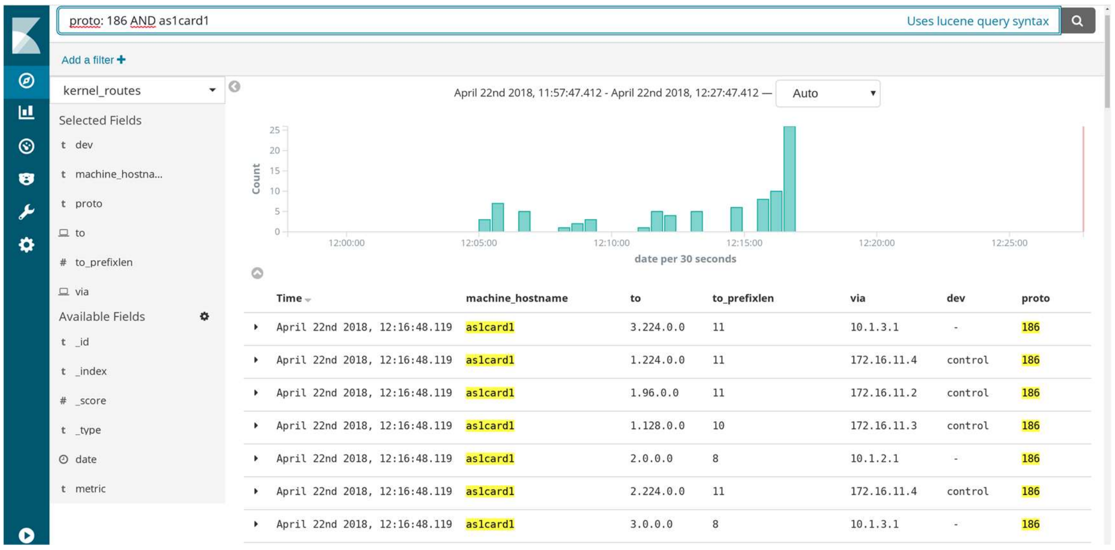


as1card1 AND prefix: 3.96.0.0

April 22nd 2018, 12:14:13.912 - April 22nd 2018, 12:19:13.912 —

Time	pathFrom	nexthop_ip	aspath	prefix	prefixLen	bestpath	machine_hostname
April 22nd 2018, 12:16:48.486	external	10.1.2.1	2 23	3.96.0.0	11	true	as1card1
April 22nd 2018, 12:16:39.548	external	10.1.2.1	2 23	3.96.0.0	11	true	as1card1
April 22nd 2018, 12:16:33.477	internal	172.16.11.2	2 23	3.96.0.0	11	-	as1card1
April 22nd 2018, 12:16:48.486	external	10.1.3.1	3 23	3.96.0.0	11	-	as1card1
April 22nd 2018, 12:16:39.548	internal	172.16.11.3	2 23	3.96.0.0	11	-	as1card1
April 22nd 2018, 12:16:39.548	external	10.1.4.1	4 2 23	3.96.0.0	11	-	as1card1
April 22nd 2018, 12:16:39.548	external	10.1.3.1	3 23	3.96.0.0	11	-	as1card1

In addition, routes on the kernel table can be queried as shown below:



proto: 186 AND as1card1

kernel_routes

Selected Fields
- t dev
- t machine_hostna...
- t proto
- to
- # to_prefixlen
- via

Available Fields
- t _id
- t _index
- # _score
- t _type
- date
- t metric

April 22nd 2018, 11:57:47.412 - April 22nd 2018, 12:27:47.412 —

Time	machine_hostname	to	to_prefixlen	via	dev	proto
April 22nd 2018, 12:16:48.119	as1card1	3.224.0.0	11	10.1.3.1	-	186
April 22nd 2018, 12:16:48.119	as1card1	1.224.0.0	11	172.16.11.4	control	186
April 22nd 2018, 12:16:48.119	as1card1	1.96.0.0	11	172.16.11.2	control	186
April 22nd 2018, 12:16:48.119	as1card1	1.128.0.0	10	172.16.11.3	control	186
April 22nd 2018, 12:16:48.119	as1card1	2.0.0.0	8	10.1.2.1	-	186
April 22nd 2018, 12:16:48.119	as1card1	2.224.0.0	11	172.16.11.4	control	186
April 22nd 2018, 12:16:48.119	as1card1	3.0.0.0	8	10.1.3.1	-	186

3. Results

This PoC demonstrates that it's possible to deploy a complex environment that can be changed in order to support security analyses of the alternatives to secure BGPv4 daemons to update routes correctly. It provides a framework to generate different environment simulations that implement security measures.

From this point, peering agreements can be defined and routers can be configured to implement them.

In addition, it provides both an easy way to generate traffic and to see the impact on terms of both kernel routing tables and accepted BGPv4 advertisements on the network. The framework also provides an interface through what the impact of certain attacks can be reviewed in both terms of routing tables and traffic volumes. Therefore, it's establishes a working environment that fulfills the requirements.

The result is like Wireshark would be when running over all bridges on the hypervisor but with some advantages: data is saved once it arrives, statistics can be computed on the fly and a friendly interface shows the path through what the traffic is traveling through. In addition, it's possible to access the history and a time-line showing changes on the network is provided.

4. Annex A

Machine Name	Management Address	Route Exchange Ifname IPv4Addr	Upstream
RPKI-RTR-Server	172.16.1.253	N/A	N/A
AS-1-1	172.16.1.11	IX_1 10.1.1.1/24 down_control 10.11.0.1/24	down 1.0.0.1/10
AS-1-2	172.16.1.12	IX_2 10.2.1.2/24 down_control 10.11.0.2/24	down 1.64.0.1/10
AS-1-3	172.16.1.13	IX_3 10.3.1.3/24 down_control 10.11.0.3/24	down 1.128.0.1/10
AS-1-4	172.16.1.14	IX_4 10.4.1.4/24 down_control 10.11.0.4/24	down 1.192.0.1/10
AS-2-1	172.16.1.21	IX_1 10.1.2.1/24 down_control 10.12.0.1/24	down 2.0.0.1/10
AS-2-2	172.16.1.22	IX_2 10.2.2.2/24 down_control 10.12.0.2/24	down 2.64.0.1/10
AS-2-3	172.16.1.23	IX_3 10.3.2.3/24 down_control 10.12.0.3/24	down 2.128.0.1/10
AS-2-4	172.16.1.24	IX_4 10.4.2.4/24 down_control 10.12.0.4/24	down 2.192.0.1/10
AS-3-1	172.16.1.31	IX_1 10.1.3.1/24 down_control 10.13.0.1/24	down 3.0.0.1/10
AS-3-2	172.16.1.32	IX_2 10.2.3.2/24 down_control 10.13.0.2/24	down 3.64.0.1/10
AS-3-3	172.16.1.33	IX_3 10.3.3.3/24 down_control 10.13.0.3/24	down 3.128.0.1/10
AS-3-4	172.16.1.34	IX_4 10.4.3.4/24 down_control 10.13.0.4/24	down 3.192.0.1/10
AS-4-1	172.16.1.41	IX_1 10.1.4.1/24 down_control 10.14.0.1/24	down 4.0.0.1/10
AS-4-2	172.16.1.42	IX_2 10.2.4.2/24 down_control 10.14.0.2/24	down 4.64.0.1/10
AS-4-3	172.16.1.43	IX_3 10.3.4.3/24 down_control 10.14.0.3/24	down 4.128.0.1/10
AS-4-4	172.16.1.44	IX_4 10.4.4.4/24 down_control 10.14.0.4/24	down 4.192.0.1/10
AS-21-1	172.16.1.211	upstream_RX_1 10.11.21.1/24 upstream_RX_4 10.14.21.1/24	down_1 1.96.0.1/11 down_4 4.96.0.1/11
AS-22-1	172.16.1.221	upstream_RX_1 10.11.22.1/24 upstream_RX_2 10.12.22.1/24	down_1 1.224.0.1/11 down_2 2.224.0.1/11
AS-23-1	172.16.1.231	upstream_RX_2 10.12.23.1/24 upstream_RX_3 10.13.23.1/24	down_2 2.96.0.1/11 down_3 3.96.0.1/11

AS-24-1	172.16.1.241	upstream_RX_3 10.13.24.1/24 upstream_RX_4 10.14.24.1/24	down_3 3.224.0.1/11 down_4 4.224.0.1/11
CLIENT-21-1	172.16.1.111	upstream 1.96.0.2/11	N/A
CLIENT-21-4	172.16.1.141	upstream 4.96.0.2/11	N/A
CLIENT-22-1	172.16.1.113	upstream 1.224.0.2/11	N/A
CLIENT-22-2	172.16.1.123	upstream 2.224.0.2/11	N/A
CLIENT-23-2	172.16.1.121	upstream 2.96.0.2/11	N/A
CLIENT-23-3	172.16.1.131	upstream 3.96.0.2/11	N/A
CLIENT-24-3	172.16.1.133	upstream 3.224.0.2/11	N/A
CLIENT-24-4	172.16.1.143	upstream 4.224.0.2/11	N/A

5. Bibliography

"BGP Routing Table Analysis Reports." 2018. <http://bgp.potaroo.net/>.

"IANA - IPv4 Address Space." 1998. <https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml>.

"IRRToolSet." 2002. <https://github.com/irrtoolset/irrtoolset>.

"JohnTheRipper." 1996. <https://github.com/magnumripper/JohnTheRipper>.

"Linux Kernel Documentation." 1991. <https://www.kernel.org/doc/Documentation/>.

Mutually Agreed Norms for Routing Security. 2014. <https://www.manrs.org>.

"RFC1105 - A Border Gateway Protocol (BGP)." 1989.

"RFC1163 - A Border Gateway Protocol (BGP)." 1990.

"RFC1267 - A Border Gateway Protocol 3 (BGP-3)." 1991.

"RFC1337 - TIME-WAIT Assassination Hazards in TCP." 1992. <https://www.ietf.org/rfc/rfc1337.txt>.

"RFC1654 - A Border Gateway Protocol 4 (BGP-4)." 1994.

"RFC1771 - A Border Gateway Protocol 4 (BGP-4)." 1995.

"RFC2385 - Protection of BGP Sessions via the TCP MD5 Signature Option." 1998.

"RFC2827 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." 2000.

"RFC3013 - Recommended ISP Security." 2000.

"RFC3704 - Ingress Filtering for Multihomed Networks." 2004. <https://tools.ietf.org/html/rfc3704>.

"RFC4012 - Routing Policy Specification Language next generation (RPSLng)." 2005.

"RFC4271 - A Border Gateway Protocol 4 (BGP-4)." 2006.

"RFC4272 - BGP Vulnerability Analysis." 2006.

"RFC5082 - The Generalized TTL Security Mechanism (GTSM)." 2007.

"RFC5925 - The TCP Authentication Option." 2010.

"RFC5961 - Improving TCP's Robustness to Blind In-Window Attacks." 2010. <https://tools.ietf.org/html/rfc5961>.

"RFC6192 - Plane, Protecting the Router Control." 2011. <https://tools.ietf.org/html/rfc6192>.

"RFC6480 - An Infrastructure to Support Secure Internet Routing." 2012.

"RFC6483 - Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)." 2012.

"RFC6810 - The Resource Public Key Infrastructure (RPKI) to Router Protocol." 2013.

"RFC6811 - BGP Prefix Origin Validation." 2013.

"RFC7196 - Making Route Flap Damping Usable." 2014. <https://tools.ietf.org/html/rfc7196>.

"RFC7454 - BGP Operations and Security." 2015.

"RFC7715 - Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)." 2016.

"RFC791 - INTERNET PROTOCOL." 1981. <https://tools.ietf.org/html/rfc791>.

"RFC7947 - Internet Exchange BGP Route Server." 2016.

"RFC8205 - BGPsec Protocol Specification." 2017.

Robert Lychev, Michael Schaipira & Sharong Goldberg. "Rethinking Security for Internet Routing." 2016.

"RPKI-RTR-Server." 2018. <https://github.com/RIPE-NCC/rpki-validator-3>.

"TCPMD5 Signature - Socket Programing Examples on Linux." 2015. <https://criticalindirection.com/2015/05/12/tcp_md5sig>.

"Ubuntu - Kernel Security Settings." 2006. <https://wiki.ubuntu.com/ImprovedNetworking/KernelSecuritySettings>.

Vivancos, José María Foces. "Rehtse." 2016. <https://github.com/JmFoces/Rehtse>.