Master Thesis
Securing the Border Gateway Protocol BGPv4

Máster Interuniversitario de Seguridad de las TIC

Universitat Autònoma de Barcelona

Universitat Rovira i Virgili

Universitat Oberta de Catalunya

Advisor: Joan Borrell Viader

Student: José María Foces Vivancos

June 2018

1. Acknowledgements

I would first like to thank my master thesis advisor, professor at Department of Information and Communications Engineering (dEIC) at Autonomous University of Barcelona, Joan Borrell Viader. For his constant support and advice on how to proceed on each step.

I would also like to thank the experts who contributed in any way to the success of this project:

- Expert in network security at Telefónica de España, Sergio Martinez Mourelle
- Cybersecurity Expert Consultant at Telefónica de España, José María Lopez Nieto
- Cybersecurity Digital Monitoring Consultant at Telefónica de España, Enrique de la Hoz
- Head of Hacking Team at Telefónica de España, Jose Hermida Prado
- Head of CSIRT-TE at Telefónica de España, Rafael Montero Montero
- Head of Network Security Design Department at Telefónica de España, Raquel González Cabello

Finally, I must express my very profound gratitude to my parents, Maria Elisa Vivancos Cerezo and José María Foces Moran, my girlfriend, Ester González Sánchez and my good friend, Jose Ángel González Andrés, for providing constant feedback, support and encouragement at every step through these months of hard work.

This accomplishment would not have been possible without them.

2. Executive Summary

Nowadays the Internet is involved in almost every task we perform. There are a lot of technologies that allow it to work but the Border Gateway Protocol (BGP) is a critical point. This protocol allows networks to share reachability information. Therefore, allowing hosts in different networks to exchange information between each other. The result is mesh of inter-connected networks, the Internet. The protocol has evolved since its' first release on 1989, the latest (fourth) version was released on 2006. But it does not provide any protections by itself and nowadays it runs the core of Internet's routing system. Because of that, it's good to recall: *"Not a single day goes by without dozens of incidents affecting the routing system…"* (Mutually Agreed Norms for Routing Security)

On this context, this project provides an analysis of protections surrounding the BGPv4 operation. In other to analyze protections performance in front of attacks we went through the following steps:

The first thing we did was to analyze at high level what were the most common protections implemented nowadays to secure the route exchange through BGPv4. This initial phase offered valuable information to set the objectives for the testing environment. The idea was to have a set of inter-connected networks simulating the internet. We demonstrated through several proofs of concept (PoCs) that it was possible to implement software that automatically deploys the environment. In addition, we achieved the ability to change and adapt the environment to our needs, so we added some tools to generate and measure traffic volumes on each machine in the network and changes on routing tables. It was a major landmark, since it provides a comfortable environment to analyze the performance of security measures in front of some attacks.

From this point we entered on an iterative process through security best practices stated at (RFC7454 - BGP Operations and Security). Note that not all of them could be analyzed since some of them are not implemented in the routing suites yet. Each iteration analyzed a protection. The analysis of each protection showed the performance of attacks against an unprotected environment by showing the traffic deviation impact and upgraded the automated deployment software to apply this protection on the whole environment and finally show the security improvement.

To conclude, we provided a testing environment over what tests could be performed, we attacked and improved the security of the routers composing the environment by protecting BGPv4 speakers, the routers by themselves, the BGPv4 TCP connections and finally network wide protections such as Resource Public Key Infrastructure (RPKI).

3. Index

4. Introduction

This section mainly summarizes (Foces Vivancos, Securing The BGPv4: PEC1 Schedule). It's explains, the context, the motivation, the objectives, the methodology, the schedule of this TFM. The document follows the schedule's structure.

4.1. Context

Internet routing system is currently vulnerable to known types of attacks. Security measures, to mitigate or completely defeat the impact of these attacks, are available or being designed nowadays but not all of them are used by entities connected to the network and providing Internet service.

Experts say: *"Not a single day goes by without dozens of incidents affecting the routing system…"* (Mutually Agreed Norms for Routing Security)

Attacks in the Internet's routing system take place usually. There are several public information sources supporting this statement. For example:

- "*14000 Incidents: a 2017 Routing Security Year in Review*" (Mutually Agreed Norms for Routing Security) dating from February's 26 of 2018.
- Route53 Amazon DNS Server , (Bitcoin Hijack: Routing Attacks on Cryptocurrencies) dating from April's 25 of 2018.

4.2. Motivation

From a personal point of view, I selected this project because I wanted to learn about these technologies. It provides me an extension of the knowledge acquired during the master and my professional career about computer networks while delving into protocols that I knew at higher level.

From a technical point of view: considering this context, where the internet routing system is currently vulnerable to known types of attacks, it's interesting to perform a security analysis security of BGPv4 since these attacks may have a great impact over any set of subjects exchanging information over the Internet.

4.3. Starting Point and benefits of the TFM

At the beginning I knew BGPv4 at high level but not in depth. I just had experience on Linux system administration, plus Bash and Python scripting and TCP/IP protocol stack knowledge and basic knowledge about BGP.

This project provided me useful knowledge about both BGPv4 and ways to secure the Internet's routing system.

4.4. Objectives

The main objectives are:

1. To provide a practical and up-to-date analysis of security measures stated at

(RFC7454 - BGP Operations and Security). For the subset that is supported by Quagga, Bird or Frrouting. Routing suites for Linux systems.

2. To deploy several virtualized scenarios to show the performance of protections in front of some attacks.

4.5. Methodology

The methodology is based mainly on the Scientific Method. We apply that on each mentioned protection and perform some research, state some hypothesis and verify them with practical examples of the attacks.

4.6. Schedule

The project consists of four main phases: define the state of art of BGPv4 Security, prepare the host for the virtualized environments and the software for automatic router configuration and deployment, protection analysis loop.

Finally, we write our conclusions and write the memory, create the slide show and record the demonstration video.

5. Development

This section outlines the main phases we went through these months of work.

We start with an initial analysis about BGPv4 operation and how to perform in a secure manner. This phase is called State of Art of BGPv4 security. This phase set the bases to design and set the requirements for the working environment. This analysis took a month long.

After that, we designed the network and implemented an automatic deployer in bash, plus some software developed in Python. The working environment offers a comfortable way to measure attacks' impact on both routing tables and traffic volumes. This phase took around a month long.

Over that initial environment, we start the protection analyses phase. This is an iteration loop over security guidelines exposed at (RFC7454 - BGP Operations and Security). During this iteration loop we improve the implementation of the automatic deployer by adding configuration directives to improve the security of the routers from the kernel configuration, going through the firewall and BGPv4 daemon configuration. This phase has two steps. The first one included the analysis of protections that do not involve Secure Inter-Domain Routing (SIDR) while the second one analyzes currently supported security protocols to secure the route exchanges behind the scope of SIDR.

To finish, we write our conclusions, based on the results of these previous phases.

5.1. State of Art of BGPv4 Security

On (Foces Vivancos, Securing The BGPv4: PEC1 Schedule), section 2, we analyzed the state of art of BGPv4 security by analyzing and summarizing different approaches of several information sources containing interesting information about this topic. A high-level summary of gathered information is provided below:

Defenses can be grouped in three major groups: Speaker, Session and Routing. For a detailed explanation refer to (Foces Vivancos, Securing The BGPv4: PEC1 Schedule).

5.1.1. Speaker defenses

The protection of the speaker focuses on hardening the router. Therefore, both firewall and network protocol stack behavior are the main topics. Protections against resource exhaustion attacks (for example syn-flood or Distributed Denial of Service (DDoS) attacks) or unauthorized access to management interfaces are analyzed.

5.1.2. Session defenses

The main objective of Session defenses is to provide origin guarantee of data exchanged through TCP by using the extension TCP-MD5 Signature. Therefore, this section focuses on preventing Main in the Middle (MITM) attacks .

In addition, Generalized TTL Security Mechanism (GTSM), a Time To Live (TTL) based protection mechanism to prevent the acceptance of packets from peers at a given number of hop distance, is analyzed. This protection has been grouped inside this section while not being strictly only a session protection mechanism.

5.1.3. Routing defenses

This section contains the defenses that can be applied on BGPv4 speakers to filter or modify received and sent advertisements. In a nutshell, these defenses control the router behavior. For example, whether to accept or deny a received route advertisement and whether to advertise or not a route to a certain network. These defenses are:

Prefix Filtering

This defense controls the IP prefixes that can be accepted and announced by ASes on BGP peering. There are two alternatives to implement this.

The first one, a short-term solution, is a static filter list, a whitelist that is configured on each router to manage advertisements made or received.

The second, is a long-term solution called SIDR. SIDR offers one service called RPKI, oriented to provide origin validation of advertisements and therefore, the capability to filter invalid advertisements. This defense protects against the following attacks:

1. ***Prefix hijack***: the same technique employed before but, in this case, it won't be able to attract the 100% of the traffic destined to this domain.

2. ***Sub-prefix hijack***: remember that longest prefix matching is used to get the best route for a network. When an AS advertise to all its neighbors that it has the shortest path to a given CIDR that does not belong to it and this domain is sub-prefix of a prefix in charge of another AS. If it's successful it will be attracting all traffic from its neighbors to this domain.

3. ***Route leak***: in a route leak the attacker violates the agreed export policies by advertising legitimate route to many of its neighbors, some of them should know that path but others do not. The impact produced is that the ASes involved in the path will attract more traffic to the prefix that they are expected to.

Note that the only restriction RPKI provides is that the last Autonomous System (AS) in the path has to be the one who owns the prefix. This protection is supported by routing suites like Quagga or Frrouting. Nokia, Cisco and Juniper routers also support it.

Topology Validation

BGPv4 does not provide a way to validate the network topology and therefore, discard advertisements that does not match with it. This defense is implemented by Secure Origin BGP (soBGP), that uses a similar PKI infrastructure as RPKI does. The PKI usage is different since the purpose of this defense is to validate the presence of links (direct connections between ASes). This defense protects against the *One-hop hijack*: the attacker can advertise a route meaning that traffic can go through it to the destination prefix (in charge of the legitimate AS).

Unfortunately, soBGP is a good known proposal but it's not implemented in the routing software suites.

Path Validation

As stated at (Rethinking Security for Internet Routing), this is a "gold standard" for BGP Security. Since the protocol does not provide a way to validate AS paths and discard advertised routes that do not match with export policies of ASes involved. There are several proposals to implement this. BGPSEC has the most traction and is getting standardized. It requires RPKI and each AS on the route append its signature to BGPv4 messages. Without this protection, the network is exposed to an *Announce an unavailable path attack*: even when the path to the destination prefix is valid (in terms of connectivity, there are links between all routers the full path), there may be another restriction, for example, one of the intermediate AS in the path does not want to forward traffic to the destination prefix. Therefore, in this case it does not announce this route. The attacker may want to announce this route to all his neighbors. Therefore, in certain circumstances, for example when the advertised path is shorter than the legitimate one. The attacker achieves to get a portion of the traffic destined to the given prefix.

The main drawback of this security measure is that the computational overhead involved would require routers with dedicated cryptographic processors and nowadays they lack of them.

5.2. Hypervisor and Environment Final Setup

This section defines the final network structure base that we used to perform the practical analyses of the protections in front of some attacks.

For technical details, please refer to (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2), that extends this section by providing software configuration details and compilation instructions.

The environment is composed by 22 routers, 8 network clients and a Resource Public Key Infrastructure – Router to Router (RPKI-RTR) Cache 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 improvement of security measures implemented to secure route exchanges through BGPv4.

5.2.1. Previous Work

Before creating this environment, we proved that was possible to deploy complex networks on the hypervisor. This initial PoC resulted in four routers of four ASes (1,2,3 and 4) and used FRrouting BGP available implementation. It was 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.

On the final deployment, the routers and any machine in general needed around 200MB of RAM.

5.2.2. 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 networks composed of 4 multi-homed ISPs.
        iii. Tier 3 networks composed by two 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 advertisement 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.

5.2.3. Environment Definition

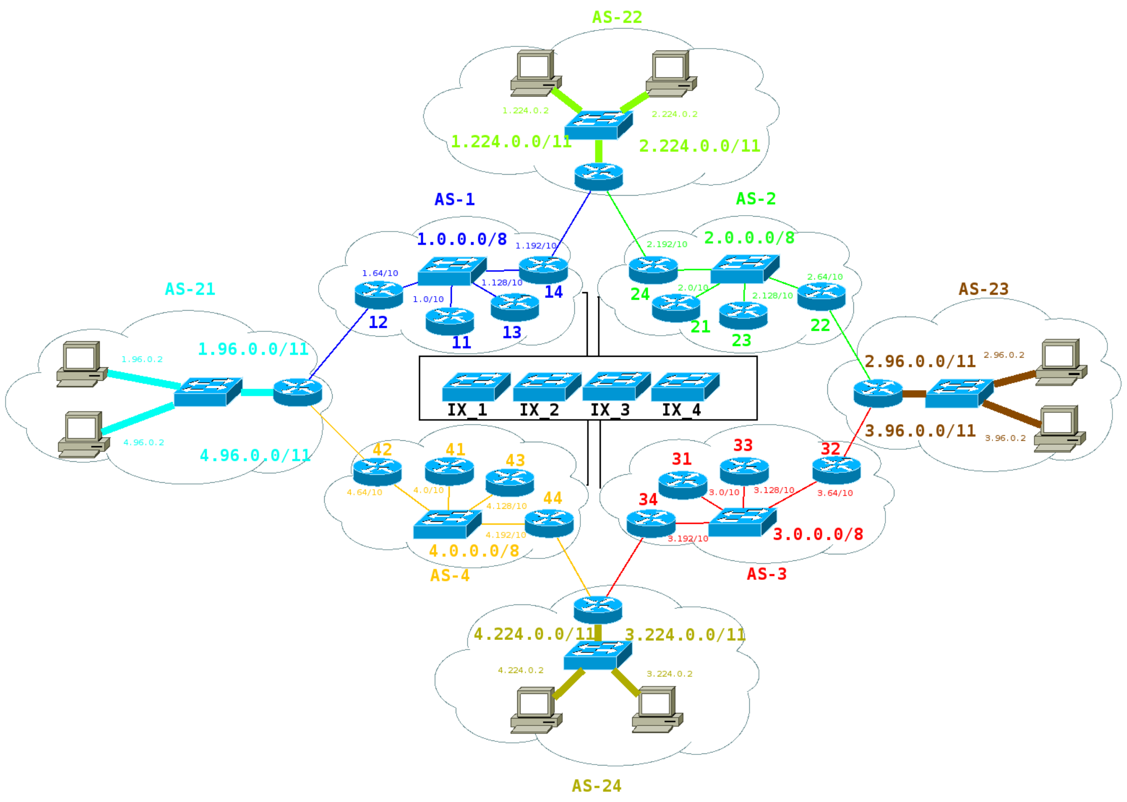
The testing environment is composed of 29 machines. All of them can be accessed through the management network 172.16.1.0/24.

Addresses are outlined on the Annex-A of (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2).

The environment is composed of several parts. A summary is provided below. For technical details please, refer to (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2).

The network is composed by several switches, simulating LAN segments between routers. There are four Tier 1 ASes, four Tier 2 ASes and 8 Tier 3 network clients. Tier one ASes are composed by four routers each, while Tier 2 just have one, connecting with two upstream ASes. Each Tier 2 has two clients behind. There is also a RPKI-RTR Cache server connected with all machines in the network. Note that the latter is not shown in the diagram. Each Tier AS is numbered from 1 to 4 and respectively, they own address spaces 1.0.0.0/8, 2.0.0.0/8 and so on. They divide this address space in 4 parts, each of them offered by one of each 4 routers as follows: 1.0.0.0/10, 1.64.0.0/10 and so on.

The following diagram shows the network topology:



5.2.4. Hardware

We deployed the whole environment on a six core i5-8600k.

On the beginning it had up to 16GB of RAM but finally we decided to upgrade it to 32GB to be sure we can make use of more memory if required (just in case). Some RPKI validator and cache servers' versions written in Java needed a lot of RAM to work.

We focused from the start on reducing the virtual machines resources at the minimum. On an initial phase it was enough to have 128MB of ram per machine but after upgrading them with some utilities they needed up to 200MB.

We used a 500GB SSD disk to speed up environment deployments.

5.2.5. Software installed.

The hypervisor runs a (Debian) 9 with kernel version 4.9. It 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 the following software installed Iperf, Netcat, Nginx and a Route Table Watcher. The latter is a simple Python script that monitors changes on the routing tables (both kernel's and BGP route tables) and send them to the Elasticsearch database.

We use Frrouting, a fork of Quagga as routing software suite.

We use Nftables as firewall. It's the replacement for {Ip,Ip6,Arp,Eb}tables.

All of them have been integrated with Systemd service manager.

RTRLib and Frrouting

In order to support RPKI-RTR protocol and therefore, to 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
4. Install the debian packages

For technical details on this process, refer to (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2).

RPKI-RTR-Server

The package (RPKI-RTR-Server) has been installed by converting the CentOS rpm package. For details about the installation process, please refer to (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2).

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.

This file defines Prefix advertisement authorizations for this testing environment. Therefore, we adapted it to our needs and environment structure.,

Iperf, Netcat, Nginx

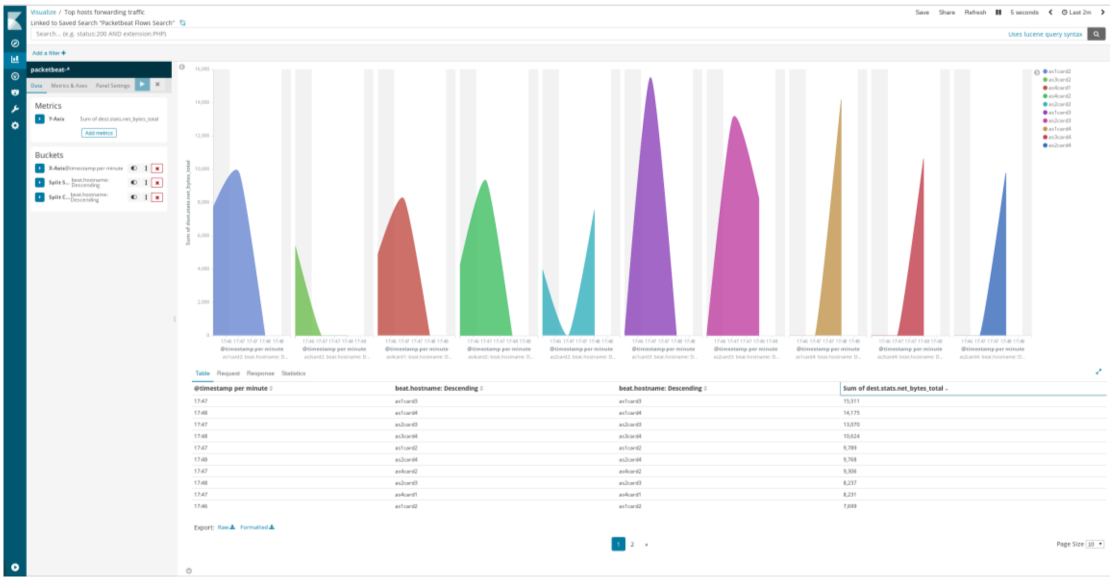
All of them are deployed as servers, so any machine in the network can send and receive traffic to and from them. They act as servers to be able to generate bi-directional traffic over the net.

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

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"

@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

Note that each color identifies a machine.

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:

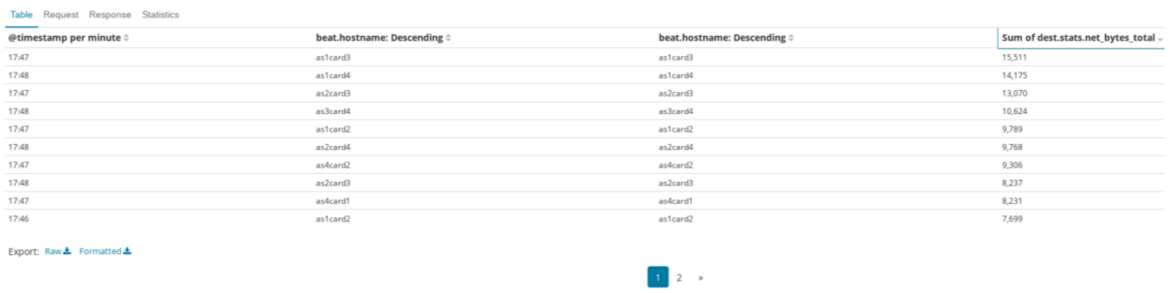
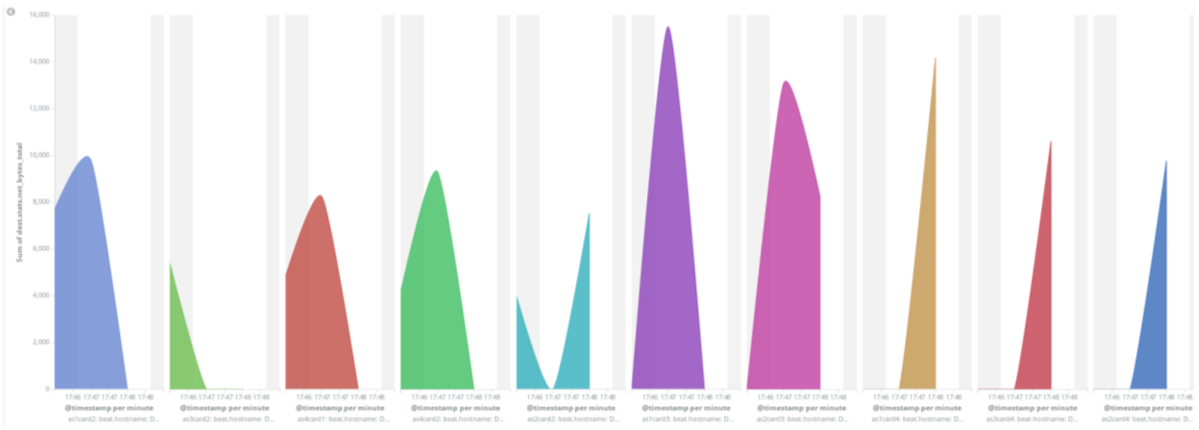


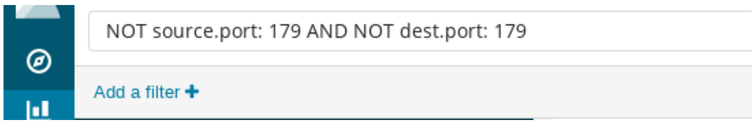
Table
Request
Response
Statistics

@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

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:

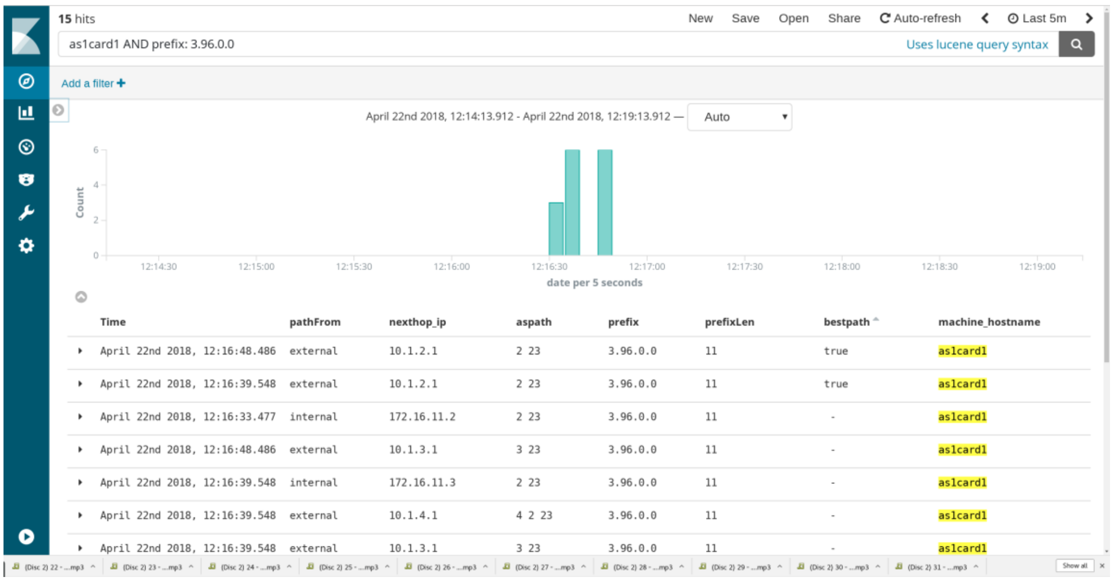


NOT source.port: 179 AND NOT dest.port: 179

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:



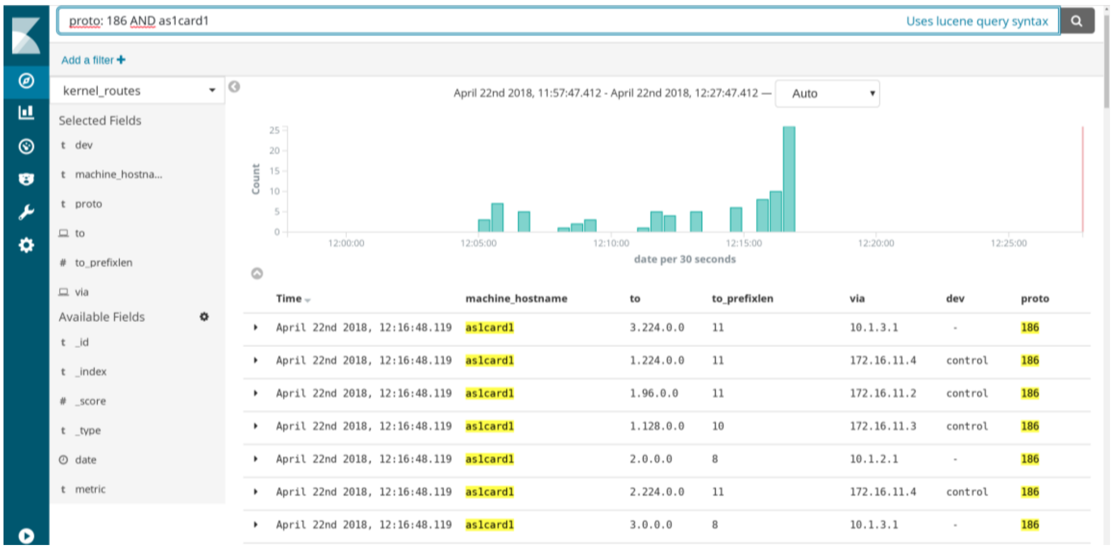
15 hits

as1card1 AND prefix: 3.96.0.0

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

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:



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

5.2.6. Results

This set of PoCs demonstrated 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 on the network, and a framework to analyze it. Therefore, it's possible to evaluate the impact of attacks over both traffic flow paths and routing tables.

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 the traffic is traveling through. In addition, access the history and a time-line showing changes on the network is provided.

5.3. Protections Analyses – Phase 1

This section contains the results of the analyses of the most popular defenses implemented currently to protect BGPv4 speakers. This phase analyzes the protections out of SIDR scope, until RPKI is used.

The environment used is exactly the environment defined at (Foces Vivancos, Securing The BGPv4: Working Environment).

In particular cases, the configuration changed to adapt to the needs of testing specific security mechanisms.

5.3.1. Objectives

The main objective is to illustrate the security improvement provided by protections and configurations explained at (RFC7454 - BGP Operations and Security), in front of some attacks.

While doing it, the automated generation of this environment is improved to integrate the explained protections. For each security measure a new environment is generated. This way the security of the whole network is improved step by step.

Defenses

Defenses can be divided in three groups:

- Speaker defenses: that involves security measures implemented at the router to protect itself.
- Session defenses: that group the protections implemented at transport or network layers to secure route exchanges.
- Routing defenses: that is composed by policies that support the decision to whether update the routing table or not

5.3.2. Speaker

In this section we explain security measures implemented to harden the speaker. The level of detail is low. Considering TFM objectives, no attacks are performed against Linux Kernel or Firewall.

For technical details, please refer to (Foces Vivancos, Securing The BGPv4: PEC2 Security Analyses - Part 1).

The main topics are Linux Kernel Networking and Firewalling and main information sources to build this section have been (Linux Kernel Documentation) and (Ubuntu - Kernel Security Settings).

Linux Kernel Networking

This section defines kernel configuration applied to prevent some attacks that would cause both router malfunction or network issues.

ICMP

net.ipv4.icmp_echo_ignore_broadcasts has been enabled to avoid answering to ICMP echo requests when destination address is broadcast. Therefore, that prevents anyone from using this device to perform source address-based DDoS attacks. At least for ICMP based ones.

net.ipv4.icmp_ratelimit & net.ipv4.icmp_ratemask respectively, they take values of 20 and 88090. Both settings combined, limit the rate of ICMP messages that may be sent in one second. The effect is that it won't send more than 5 ICMP messages per second of each of the following types: Echo Reply, Destination Unreachable, Source Quench, Time

Exceeded, Parameter Problem, Timestamp Reply, Information Reply. That sets a maximum of 35 ICMP packets per second.

net.ipv4.icmp_ignore_bogus_error_responses avoid logging bogus ICMP error responses. This would lead to fill up the disk with useless log entries.

net.ipv4.conf.all.secure_redirects it's enabled by default. But it's good to ensure activation, since it prevents hijacking of routing paths, critical on this context.

net.ipv4.conf.all.shared_media it's enabled by default. However, it's good to ensure activation since it disables secure redirects.

TCP

net.ipv4.tcp_synack_retries This is set to 2 (default is 5). The purpose is double, avoid sending TCP SYNACK responses during a SYN Flood attack and to limit the time that resources supporting this connection establishment remain allocated.

net.ipv4.tcp_syn_retries has been limited to 2 (instead of default value of 6) despite not having an impact directly on security. This way we can assume that we have a network issue, if two SYNs have no response.

net.ipv4.tcp_syncookies is enabled by default. Improves the behavior of the system when the TCP SYN Queue is full, by not attempting to introduce another entry but discarding and delaying the resource allocation to the reception of the TCP-ACK. After that, the system can rebuild the SYN Queue entry from TCP's sequence numbers.

net.ipv4.tcp_rfc1337 is disabled by default. It changes the behavior of the system when closing TCP connections and prevents from TCP TIME-WAIT state hazards.

net.ipv4.tcp_max_syn_backlog limits the size of the SYN Queue. Default value is 8192. Considering limitations of simulations, this value is set to 1024, that is enough for these test cases.

net.ipv4.tcp_window_scaling enabled by default. However, it's recommended to disable this since it enlarges the TCP Window size, and therefore, it eases TCP-RST attacks. Concretely, this is critical on this context, since BGPv4 uses long lived connections. It does not provide full protection against this kind of attack but, it may be hard to successfully execute it.

Firewall

This section defines firewall configuration applied to mitigate or prevent certain types of attacks in concrete contexts.

As explained in (RFC7454 - BGP Operations and Security) and (RFC6192 - Plane, Protecting the Router Control), access control lists must be defined to protect the management and route exchange (AKA control) networks. Both are usually referred as planes.

In addition, (RFC3704 - Ingress Filtering for Multihomed Networks) and (RFC2827 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing) have been considered.

Therefore, the first thing that should be considered to prevent unauthorized access or service disruption from forward to management or control planes is to decide what is legitimate traffic. Further steps may involve rate limiting and limiting maximum TCP connections opened from the router on the control and management planes.

For this concrete case study, it's trivial to filter legitimate traffic, since all routers speak BGP through (IANA - IPv4 Address Space) private ranges. Rate limits and maximum connections opened have been applied but just for testing purposes. Attacks won't be executed against these defenses.

To implement these ACLs, we use Nftables, the replacement for Iptables. The configuration is comfortable and easy to understand. The project is still on development. The version used is 0.8.3 in conjunction with Linux Kernel version 4.16.0. Both allow to set up FIB (Forwarding Information Base) queries to implement reverse path on the firewall. Performing this task on the firewall improves reverse path filters flexibility. It's possible to configure reverse path filtering directly over kernel's configuration (net.ipv4.conf.<iface>.rp_filter) but it lacks flexibility. This way more precise filters can be applied.

Our example firewall configuration defines the following entities that are used on the ACLs:

- Allowed port sets by interfaces. Control (179, BGPv4) and Forward (2115 (netcat), 5001 (iperf), 80 (nginx))
- Trusted interfaces composed by lo and management.
- Blackhole ip set, that matches addresses belonging to multicast or special purpose CIDR domains: 0.0.0.0/8, 127.0.0.0/8, 169.254.0.0/16, 192.0.2.0/24, 240.0.0.0/5, 248.0.0.0/5
- Control ip set, that matches any ip inside below private CIDR domains used to perform route exchange: 10.0.0.0/8, 172.16.0.0/16

We implement the following policies. Note that some of them are not as strict as they should be to ease the use of the working environment.

- Input filters. Default policy is drop. Anything not stated is disallowed:
    - Any packet ingressing through management or lo interfaces.
    - Accept any already established connection.
    - Accept ICMP packets for rates behind 10 packets per second.
    - Accept maximum 10 new connections to forward port set per second.
    - Accept maximum 10 new connections to control port set per second.
    - Accept packets with ttl value of 1 to respond to traceroutes.

- Forward allows everything but packets destined to blackhole ip set.
    - Note that reverse path filter is implemented here to protect the management plane. Forbid traffic that comes to the router when destination address does not match interface address where it's arriving.
- Any output is accepted.

5.3.3. Session

This section is mainly oriented to show protections' performance against certain attacks, applicable to communication channels between peers exchanging routes through BGPv4.

As stated at (RFC7454 - BGP Operations and Security), the following security measures should be applied to protect BGPv4 Sessions: Transport Layer and GTSM. While the latter can be considered complementary, in certain cases, with speaker protections explained on previous section. It provides additional barrier that insert another protection layer to the BGPv4 speaker.

Transport Layer

BGP works over TCP. Therefore, successful attacks to TCP connections are also applicable to BGP Sessions, since the protocol is not oriented to protect sessions by itself.

There are two TCP extensions to provide origin guarantee of packets exchanged through a TCP connection established between two peers at transport layer. They are TCP-MD5 and TCP-AO, respectively defined at (RFC2385 - Protection of BGP Sessions via the TCP MD5 Signature Option) and (RFC5925 - The TCP Authentication Option). While TCP-AO provides stronger protection, the most popular is TCP-MD5 since TCP-AO is not supported by the equipment deployed currently. For example, Linux lacks support for TCP-AO nowadays.

Considering that, this section only focuses on TCP-MD5 extension.

TCP-MD5 Signature

TCP-MD5 Signature is an extension to the TCP protocol that provides origin guarantee to both peers communicating through it. It requires a password to be set and used at both connection's sides.

All routers on the environment have been configured to use this TCP extension when communicating with other peers.

A weak password (JMFVTFM) has been set to speed up tests performed against this security measure, since it involves cracking.

As stated on (RFC2385 - Protection of BGP Sessions via the TCP MD5 Signature Option), the input to the hash function (MD5) is (obviously, order means):

1. the TCP pseudo-header (in the order: source IP address, destination IP address, zero-padded protocol number, and segment length)

2. the TCP header, excluding options, and assuming a checksum of zero
3. the TCP segment data (if any)
4. an independently-specified key or password, known to both TCPs and presumably connection-specific

Therefore, an attacker, that have access to the communication media, can sniff traffic and after that, perform a brute force attack to discover the 4th part of the input to the MD5 hash function.

To show the behavior on the testing environment, TCPMD5 Signature protection is enabled. The examples to enable TCPMD5Signature extension on a Linux client socket from (TCPMD5 Signature - Socket Programing Examples on Linux) are used.

We analyzed the socket behavior in both cases good and bad passwords. As expected, the connection is never established unless the good password is set. The TCP SYN packets are dropped at the server socket side.

The purpose of this attack is to show that is possible to brute force weak passwords and to create conscience about the use of strong ones for protecting BGPv4 Sessions.

There are several tools to perform brute force attacks. For this case the most suitable seems to be (JohnTheRipper). John is a password cracker and support several formats and it has a set of scripts that ease the conversion from almost any format to a valid input for itself.

Concretely, it has a script named pcap2john.py that extracts parts 1,2 and 3 for the MD5 hash function from a PCAP file plus the hash. The output of this script output is valid as input for John.

We took a capture using Wireshark. After that, we used pcap2john and selected one packet from the output to be cracked.

The first three parts of the input(hex-encoded) to the hash function:

Saddr	Daddr	Prot	Len	Sport	Dport	Seq	Ack	Flags	Wsize	CRC
0a010101	0a010201	0006	0030	00b3	8036	dd279a68	37e80835	c012	7210	00000000

This input is concatenated with the given password and given as input to the MD5 hash function.

In this example, we select an incremental attack, that tests all possible combinations of the tail (password used for TCPMD5 Signature) but with only upper-case ASCII characters. It would be hard to obtain a stronger key cracking it just with the CPU, but this attack can be easily parallelized and executed faster with a GPU. We cracked this example password in 3 minutes.

Attack 2: MITM

The purpose of this attack is to show the protection offered by TCPMD5 Signature against MITM attacks.

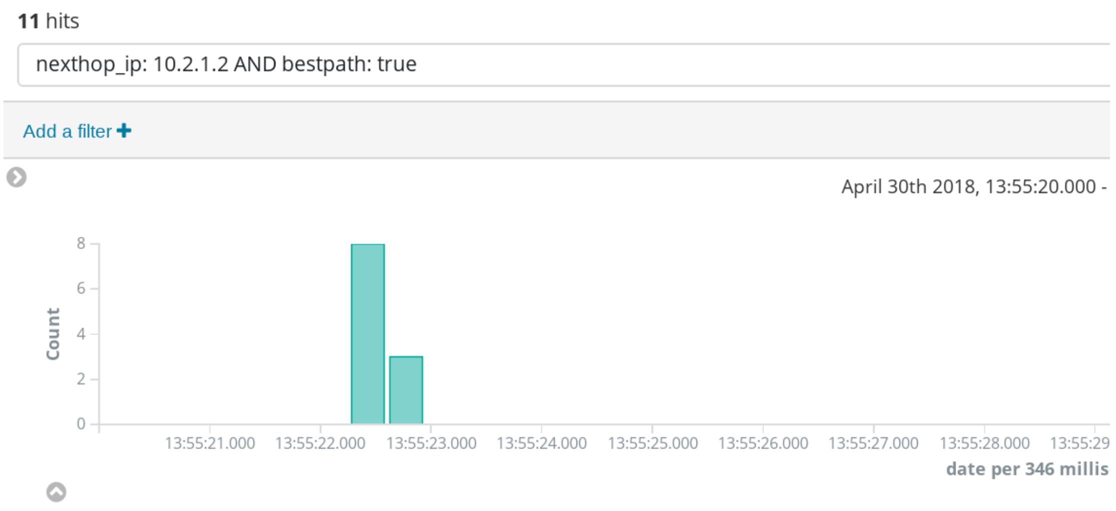
Therefore, for testing this, TCPMD5 Signature has been disabled to show the impact of a MITM attack where NEXT-HOP gets replaced and after that, enabled it back to show the correct behavior.

With TCPMD5 Signature enabled the victim router notices the MITM attack and invalidates received routes.

We reset a router three times, the first to show the normal behavior, the second to show the performance of a MITM attack without TCPMD5 Signature and finally the same attack is repeated with TCPMD5 signature enabled.

Normal behavior

As can be appreciated, 11 route updates take place upon resetting this router affecting the target next hop we will attack.


11 hits

nexthop_ip: 10.2.1.2 AND bestpath: true

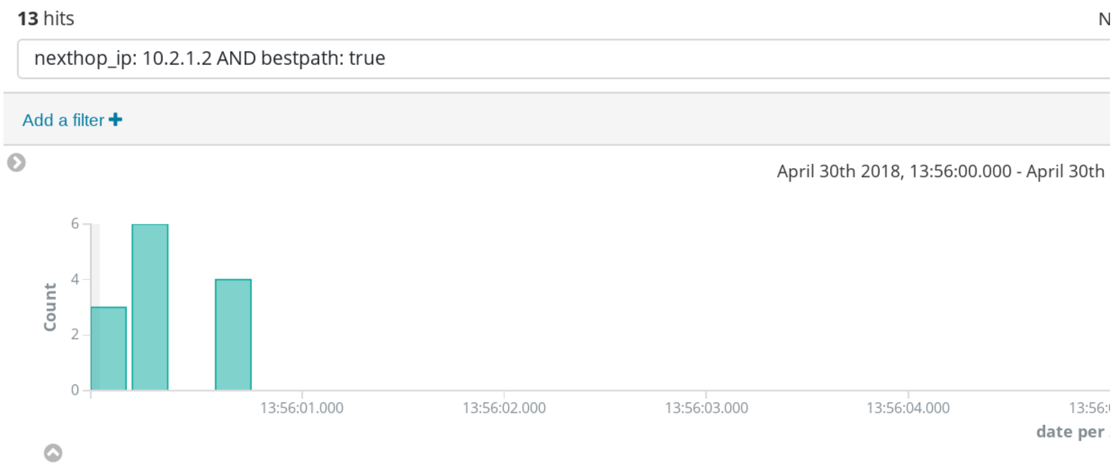
Without MD5Signature

The attacker replaces NEXT_HOP path attribute by 10.2.1.2 on any packet reaching or leaving as4card2 on IX_2 switch. Therefore, causing traffic convergence on this host.

To be able to replace certain patterns on this set of packets we use (Foces Vivancos, Rehtse). This software was implemented some time ago, but it fits perfectly for this case. It takes packets from Netfilter Queue and applies regex-based replacements on certain packages.

To simulate the attack Rehtse runs on the hypervisor.

With Rehtse running, BGPd on As4card2 is restarted again. This time a total of 13 best paths going through 10.2.1.2 are shown:


13 hits

nexthop_ip: 10.2.1.2 AND bestpath: true

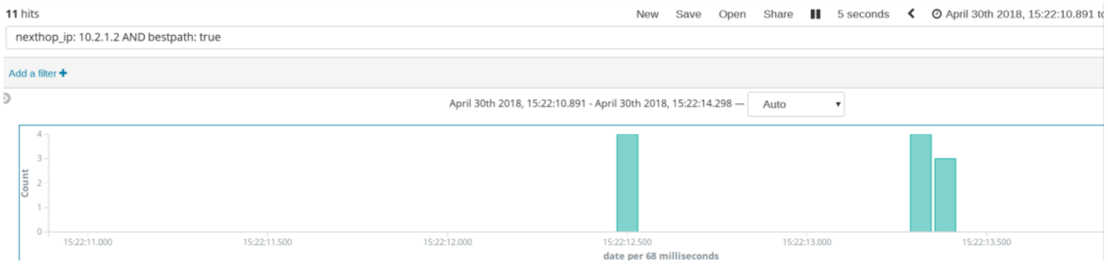
April 30th 2018, 13:56:00.000 - April 30th

Therefore, the attacker succeeded to increase traffic convergence by just tampering with traffic destined to one router.

With MD5Signature

We follow the same procedure as shown before, shutdown as4card2 enable the MITM attack and start BGPd again.

As shown below, the attacker was unable to tamper the next hop and only packets that had next-hop 10.2.1.2 outgoing from as4card2 were accepted by others in IX_2 and vice versa.


11 hits

nexthop_ip: 10.2.1.2 AND bestpath: true

April 30th 2018, 15:22:10.891 - April 30th 2018, 15:22:14.298 — Auto

On this case the routes are learned by as4card2 from other routers but not from the tampered session.

GTSM

Generalized TTL Security Mechanism, defined at (RFC5082 - The Generalized TTL Security Mechanism (GTSM)) is a technique to protect two peers exchanging IP datagrams, in general. The protection is based on IP protocol TTL field. As (RFC791 - INTERNET PROTOCOL) states at sections 3.1 and 3.2 the TTL field should be decremented by one by each processor of the packet.

This decrement is the base of traceroute utility, where TTL starts with a value of 1 and is incremented by one. So, an ICMP time-exceeded is returned to the machine tracing the route to a host for each router in the path.

Therefore, the TTL field can be used to get the hop distance between two peers. GTSM is based on that and makes the router to discard packets that do not match the distance filter established. We configure a maximum hop distance of 1.

When configured this way, it will make the router to only accept packets coming to the TCP socket of the BGPv4 daemon with a TTL of 255.

While remaining simple and cheap in terms of CPU cycles, it provides another defense layer for BGPv4 Speakers and Sessions, since it won't be possible to perform attacks against a sessions or peers connected on the same media (no routers in the middle). Unless the attacker has previously hijacked a device connected to the same media. In this case, the attacker, would make this hijacked device to rewrite TTL value with 255 and therefore be able to tamper with the session established. But given the case, one of the least desirable things he would do is to tamper with an established BGPv4 session.

For testing the impact of that configuration directive, we selected two routers. Without GTSM enabled, routers speak BGPv4 over TCP and them, over IP with TTL 1. With GTSM enabled TTL takes the value 255. That is a smart tactic, considering that default TTL values change between manufacturers and taking the upper bound because it's decremented by design.

Since attacks against IP protocol are out of the scope of the project, we showed the behavior of this implementation with different TTL values. We modified the TCPMD5 example program to send a BGPv4 OPEN message.

We demonstrated that TTLs indicating hop distance far away than configured are not allowed to become peers. For small TTLs the connection is not even established but with greater ones, the peer configured with GTSM enabled does not accept BGPv4 messages from the other side unless it sends these messages with TTL 255, in this concrete case, with adjacent hosts (hop distance of 1).

5.3.4. Routing

This section is mainly oriented to show protections' performance against certain attacks, applicable to advertised and accepted routes by BGPv4 speakers. This is one of the most critical aspects when securing BGPv4 routers.

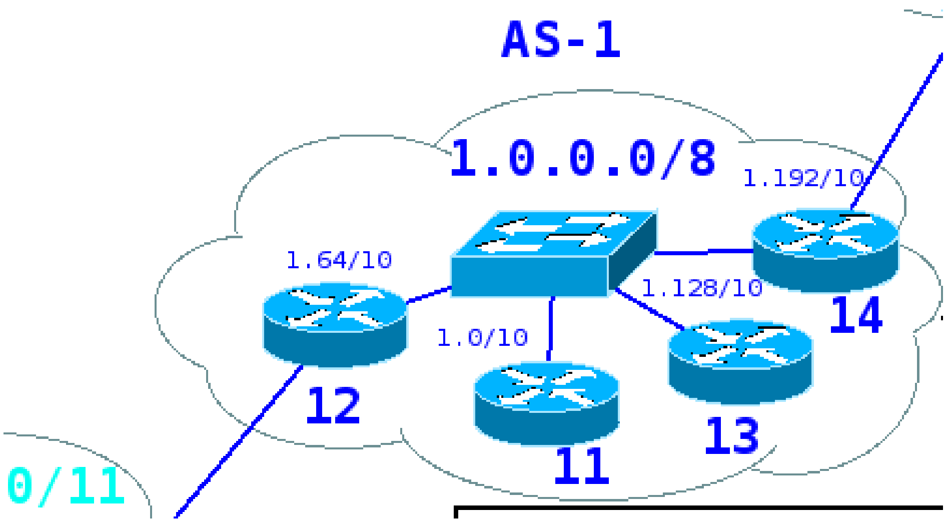
Prefix Filters

As stated on (RFC7454 - BGP Operations and Security), any special purpose or unallocated IANA prefixes should be filtered from being advertised or accepted. In addition, Regional Internet Registries filters should be considered too. The first group is easy to maintain since it's static, but the latter changes over the time and the management load is high. To ease this task there is a tool called (IRRToolSet). That works conformant with (RFC4012 - Routing Policy Specification Language next generation (RPSLng)), to exchange routing policies. With this, the network administrator can keep updated the prefixes lists and who is allowed to advertise them, but this is a wider topic and it's out of this section.

A prefix filter is composed by a network prefix and an action. Optionally, it may include [le|ge] operators. Respectively, they filter lesser or equal and greater or equal prefixes belonging to the provided one.

We configured the environment to make Tier 1 AS routers not to leak internal AS structure using prefix filters.

Regarding AS1 structure, each router provides access to a quarter of the address space 1.0.0.0/8. As shown below:



Prefix filters prevent that internal structure of AS1 is advertised outside.

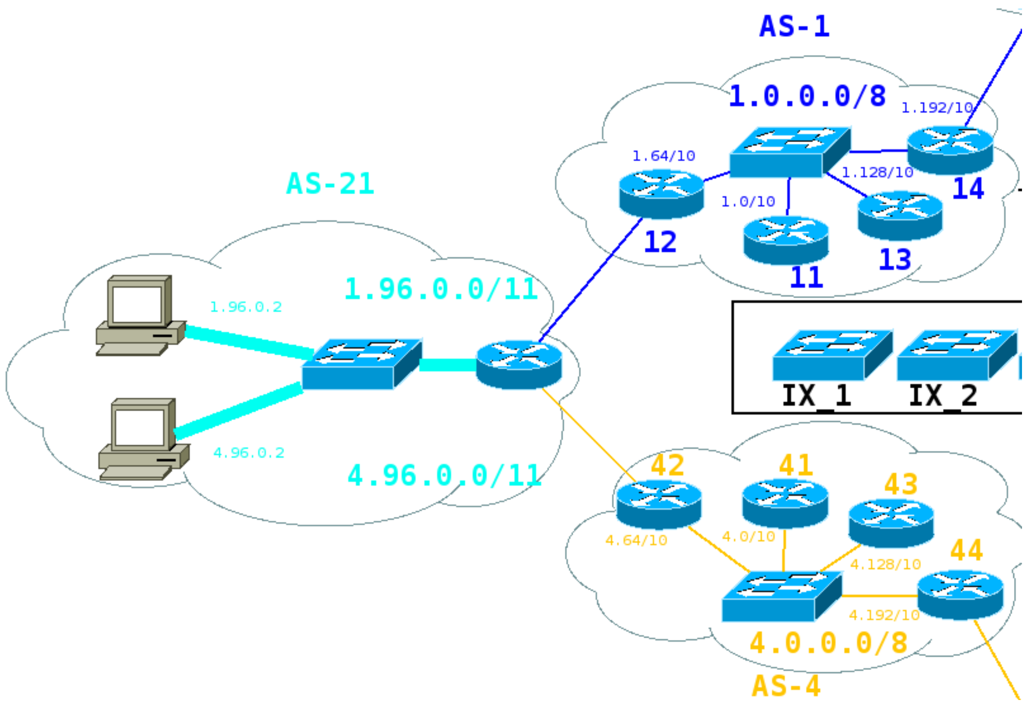
Attack: Route Leak

When configured rigorously, prefix filters prevent routers from accepting or advertising prefixes that should not. Note that this is easy to configure, but hard to maintain considering the frequency of network topology updates.

In a route leak, an AS violates agreed export policies.

To illustrate that, we select a Tier 2 AS router. It should only advertise prefixes on its own, since it would never want to transit traffic between its upstream providers.

We also introduce another configuration mistake. Tier 1 ASes do not filter correctly and leak internal structure to Tier 2 ASes.



To show the effect we select AS21's router and change the configuration deployed automatically. This configuration prevent the leak since Tier 2 ASes only advertise them prefixes.

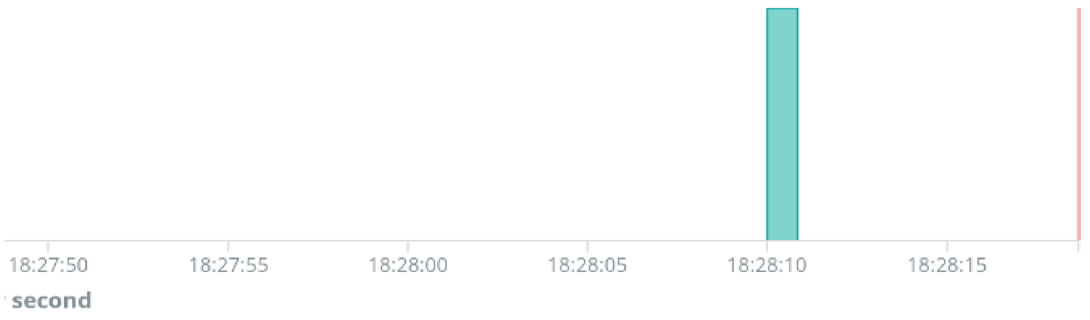
On this case, neither AS4 and AS1 know the internal structure of each other. They do not advertise them internal structure to each other, but they have a configuration mistake, they advertise it to AS21. Initially, AS21 is exporting routes as expected and just advertise its' prefixes.

Before we apply any changes, we show the behavior without tampering with AS21 export policies. After that, we show the effect that AS21 produces when violating its normal export policies.

Normal case

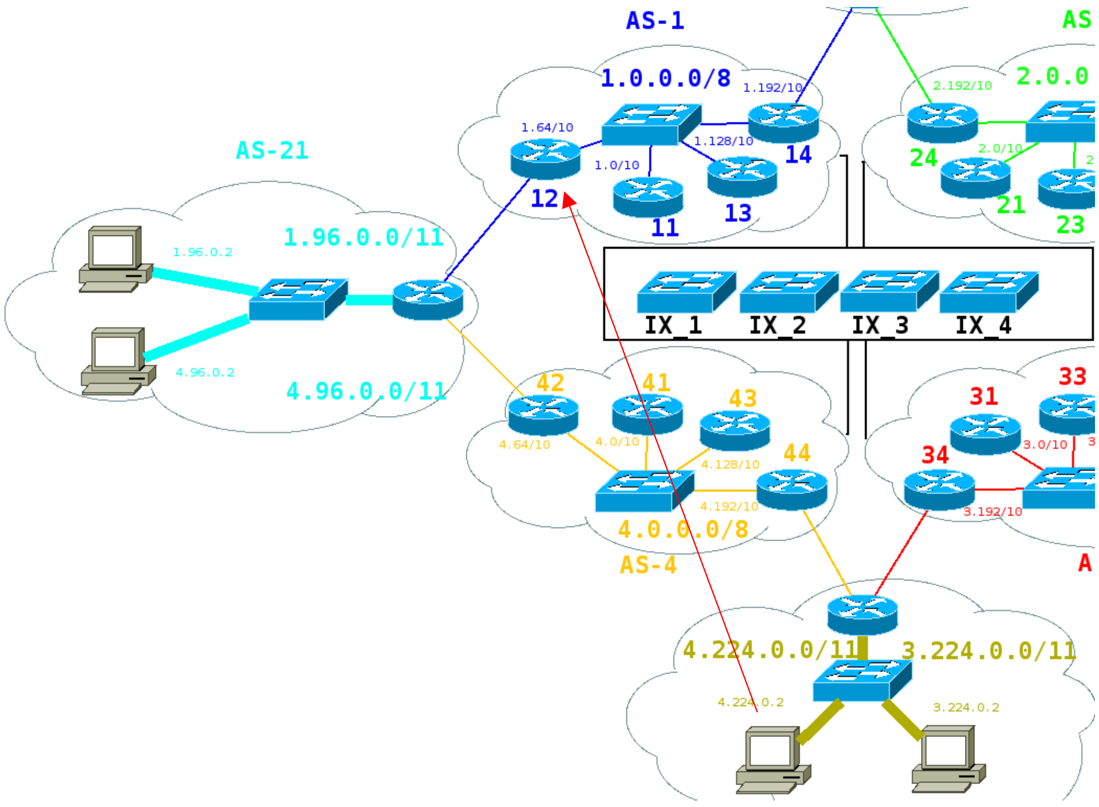
Querying BGPv4 route index history on Kibana with the filter:

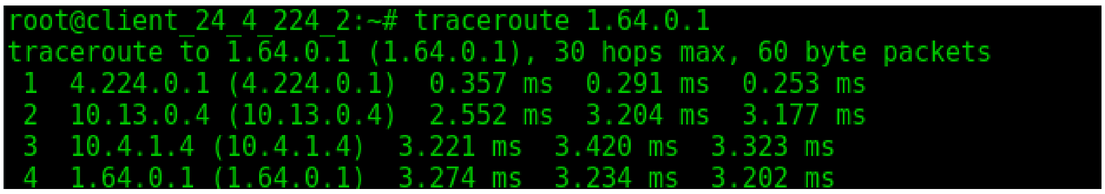
```
machine_hostname:as1card2 AND bestpath: true AND prefix: "4.0.0.0/8"
```



prefix	prefixLen	nexthop_ip	aspath
4.224.0.0	11	10.2.3.2	3 24
4.96.0.0	11	10.11.21.1	21
4.0.0.0	8	10.2.4.2	4

Tracerouting from somewhere in the network before the attack:

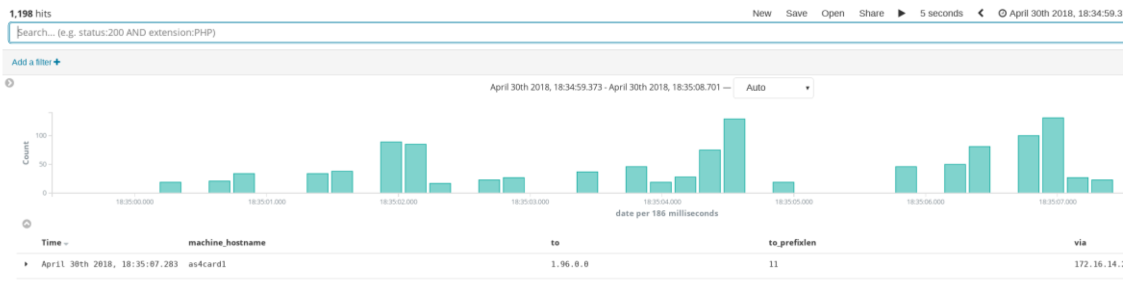



```
root@client_24_4_224_2:~# traceroute 1.64.0.1
traceroute to 1.64.0.1 (1.64.0.1), 30 hops max, 60 byte packets
 1  4.224.0.1 (4.224.0.1)  0.357 ms  0.291 ms  0.253 ms
 2  10.13.0.4 (10.13.0.4)  2.552 ms  3.204 ms  3.177 ms
 3  10.4.1.4 (10.4.1.4)  3.221 ms  3.420 ms  3.323 ms
 4  1.64.0.1 (1.64.0.1)  3.274 ms  3.234 ms  3.202 ms
```

AS21 violating export policies

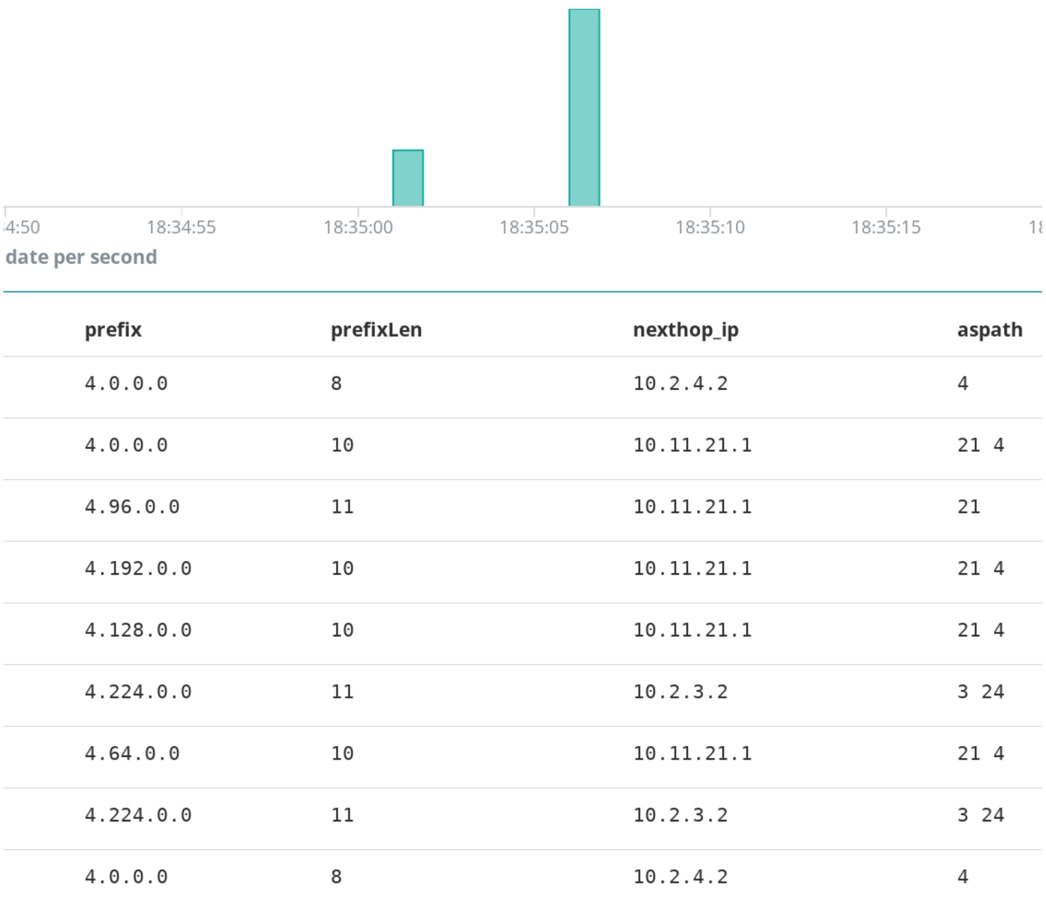
On this step, we make AS21 to violate normal export policies by disabling export filters and see the effect.

When the configuration changes, the whole network topology changes and kernel routing tables suffer near 1200 updates:


1,198 hits
New
Save
Open
Share
5 seconds
April 30th 2018, 18:34:59.3
Search... (e.g. status:200 AND extension:PHP)
Add a filter +
April 30th 2018, 18:34:59.373 - April 30th 2018, 18:35:08.701 — Auto
Count
date per 186 milliseconds

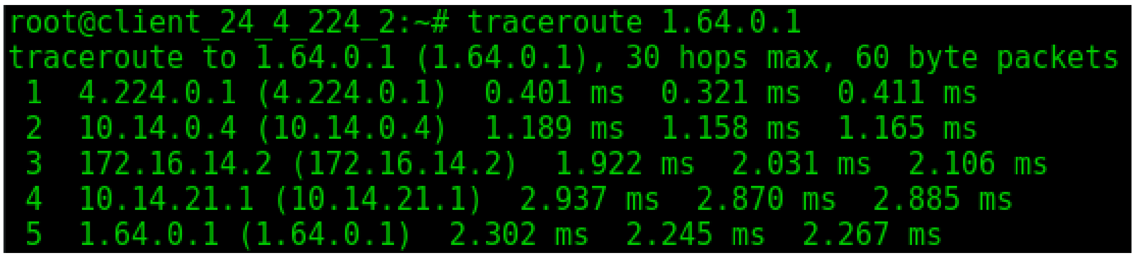
Time	machine_hostname	to	to_prefixlen	via
April 30th 2018, 18:35:07.283	as4card1	1.96.0.0	11	172.16.14.

Using the same filter on Kibana, it's possible to see that as1card2 now knows other paths to reach certain sections of 4.0.0.0/8.



prefix	prefixLen	nexthop_ip	aspath
4.0.0.0	8	10.2.4.2	4
4.0.0.0	10	10.11.21.1	21 4
4.96.0.0	11	10.11.21.1	21
4.192.0.0	10	10.11.21.1	21 4
4.128.0.0	10	10.11.21.1	21 4
4.224.0.0	11	10.2.3.2	3 24
4.64.0.0	10	10.11.21.1	21 4
4.224.0.0	11	10.2.3.2	3 24
4.0.0.0	8	10.2.4.2	4

The impact is obvious, the route from AS4 client to as1card2 has changed and now goes intra-AS4, traverses AS21 and reaches the as1card2.



```
root@client_24_4_224_2:~# traceroute 1.64.0.1
traceroute to 1.64.0.1 (1.64.0.1), 30 hops max, 60 byte packets
 1  4.224.0.1 (4.224.0.1)  0.401 ms  0.321 ms  0.411 ms
 2  10.14.0.4 (10.14.0.4)  1.189 ms  1.158 ms  1.165 ms
 3  172.16.14.2 (172.16.14.2)  1.922 ms  2.031 ms  2.106 ms
 4  10.14.21.1 (10.14.21.1)  2.937 ms  2.870 ms  2.885 ms
 5  1.64.0.1 (1.64.0.1)  2.302 ms  2.245 ms  2.267 ms
```

BGP Route Flap Dampening

This configuration directive allows to penalize routes that change frequently. This route changes that use to take place waste CPU cycles that may be used for other purposes and this, is the reason of its existence. Initial research shown that it would cause more harm than benefit and therefore, the RIPE community recommended against using it in 2006. Some years after, in 2014, researchers of Internet Initiative Japan, Internet Initiative Japan, Sproute Networks and Loughborough University shown how to make efficient use of this technique on (RFC7196 - Making Route Flap Damping Usable).

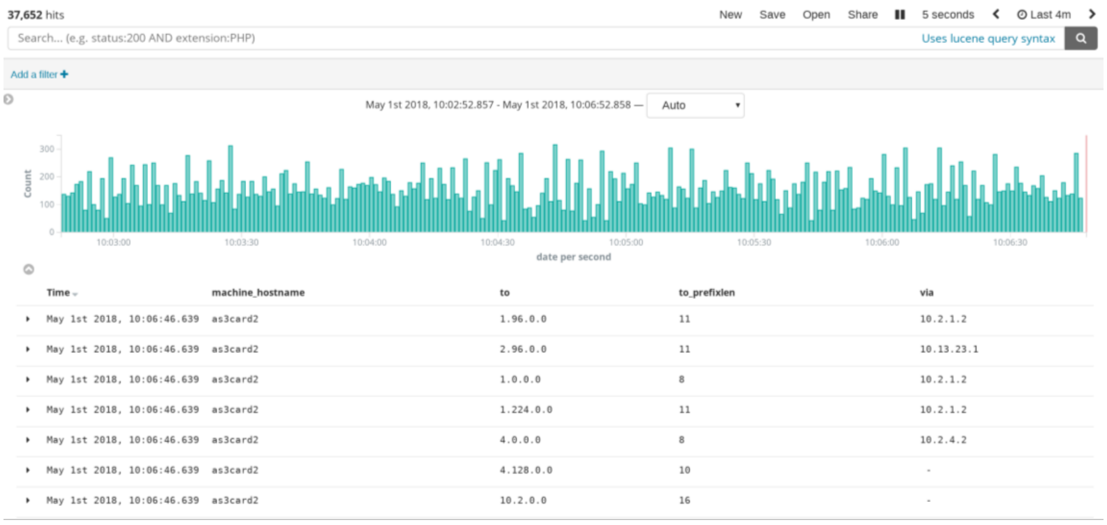
Attack: Flapping the previously explained route leak

To simulate the behavior of a flapping route we use as21card1. It will periodically advertise the route leak shown before and afterwards get back to its normal export policies.

We use a Bash & Vtysh script to generate this behavior.

That makes both as1card2 and as4card2 to make changes to them routing tables each period and to advertise these changes to them peers.

The following timeline shows the effect over kernels routing tables of the whole network and for a 4 minutes period:



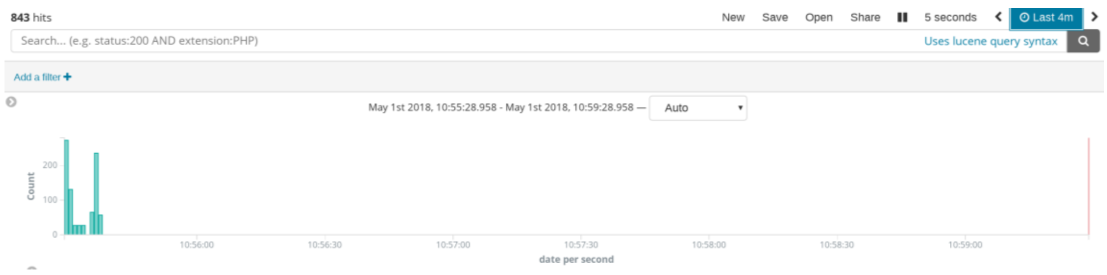
Time	machine_hostname	to	to_prefixlen	via
May 1st 2018, 10:06:46.639	as3card2	1.96.0.0	11	10.2.1.2
May 1st 2018, 10:06:46.639	as3card2	2.96.0.0	11	10.13.23.1
May 1st 2018, 10:06:46.639	as3card2	1.0.0.0	8	10.2.1.2
May 1st 2018, 10:06:46.639	as3card2	1.224.0.0	11	10.2.1.2
May 1st 2018, 10:06:46.639	as3card2	4.0.0.0	8	10.2.4.2
May 1st 2018, 10:06:46.639	as3card2	4.128.0.0	10	-
May 1st 2018, 10:06:46.639	as3card2	10.2.0.0	16	-

As it can be appreciated, that affects a lot of routers, producing changes on kernel routing tables.

To test that, we enabled route flap dampening protection on all routers as stated on (RFC7196 - Making Route Flap Damping Usable) for Cisco routers:

```
bgp dampening 15 750 2000 60
```

And repeated the attack performed on as21card1 for a 4 minutes period again:



With route flap dampening enabled, 36809 kernels routing tables updates on the whole network were prevented. Only the first 843 took place.

This configuration directive, when used correctly, prevents frequent route updates, it saves resources and provides improved network topology stability.

This is just an example, but an attacker would use other ways to cause route flaps. For example, by tampering with unsecured BGPv4 sessions, performing DoS or DDoS attacks over old routers that may be easily exhausted or just by connecting to a badly configured router and performing periodic advertisements. They would be others.

Maximum Prefixes on a Peering

This configuration directive limits the maximum routes that can be accepted from a peer.

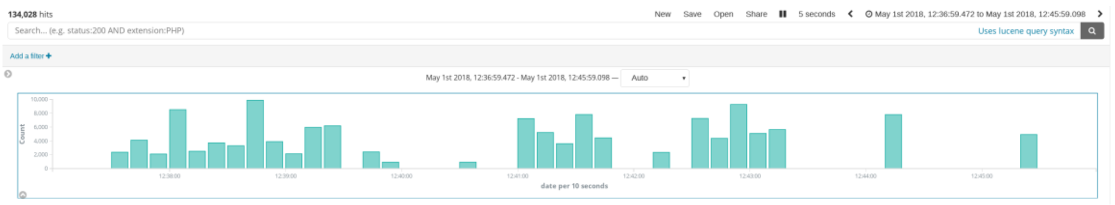
The best current practice states that maximum prefixes on a peering should be limited on both peers and upstream routers. The main objective is to protect routers memory from exhaustion.

Attack: Advertising routes for each host but for the network

To test that, we will use as21card1 again. We change the configuration, so it starts advertising a route for each host in both domains it owns. It will advertise 2 * 2^16 routes if the full iteration process finishes.

For that purpose, we use Bash & Vtysh that will generate routes for each host in some legitimate networks owned by AS21.

When running, as21card1 produced an impact of ~134000 kernel route updates in 10 minutes on the overall network.

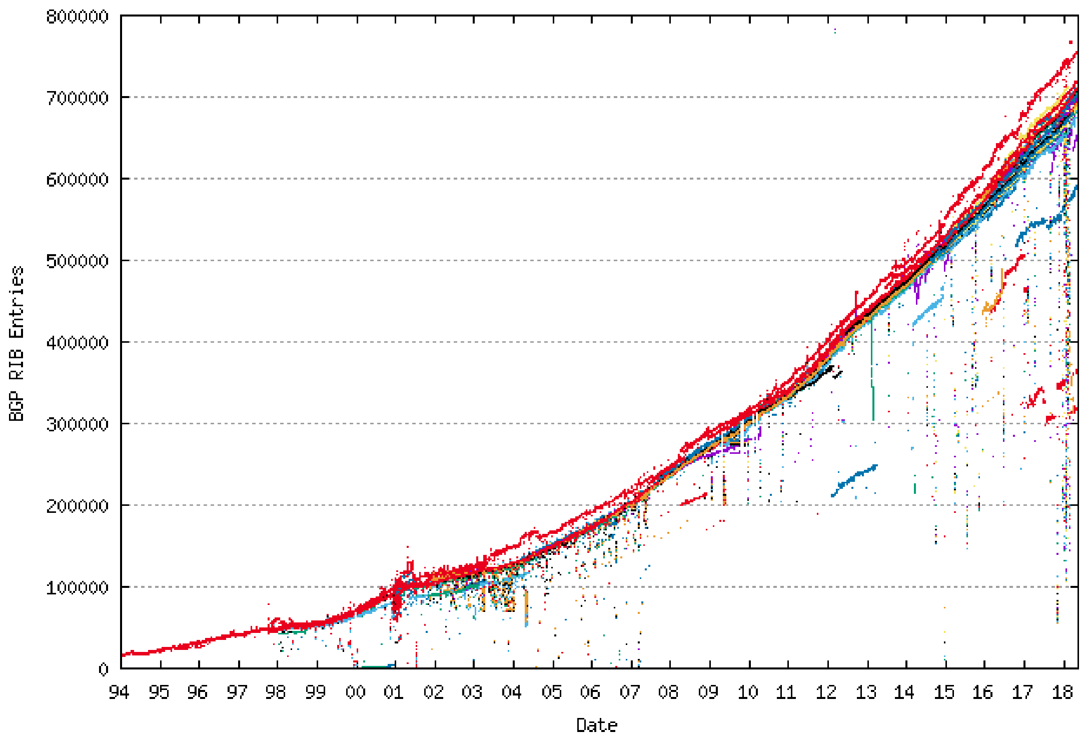


Both as4card2 and as1card2 became unresponsive for a while when they had around 20k routes respectively on them tables.

Keeping it running shows that soon they will have memory problems due to the limitations of the environment.

Around 35k routes, as1card2 and as4card2 were almost unresponsive. On this point we stop the attack. Around (63·255)·2 routes were announced. We restarted BGPd on as21card1, so other routers can get back on normal operation.

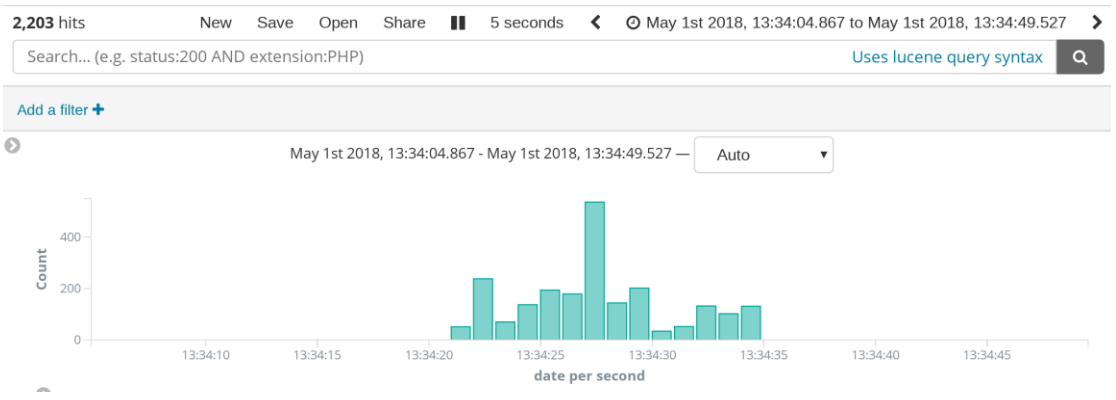
It's commensurate that a router with more resources will keep running without issues against this kind of attack. But routing table sizes can be huge on the internet. As shown at (BGP Routing Table Analysis Reports) the BGP routing table size is growing faster and nearly it will reach 800k entries for a router working with full BGP table.



Neither Quagga or Frrouting have documentation about how to configure this. But we've found that it works almost like Cisco routers.

To protect from this kind of attacks, a general limit has been applied and no more than 100 prefixes will be accepted by any router on the network. If anyone advertises more than the threshold the BGPv4 session will be restarted after 150 minutes.

The impact of the attack is limited, routers were able to keep themselves stable.



The BGPv4 sessions were restarted from as1card2 and as4card2 when more than 100 prefixes were advertised from as21card1. In addition, since they will never accept more than 100 their sessions with other peers won't be restarted and the session will be shut down so as21 will get banned by their providers for 150 minutes.

AS Path Filtering

As path filters allow to filter accepted and advertised routes if the as-path matches a given pattern.

As best practices state, this configuration directive, should be used by network administrators:

- To avoid accepting:
    - Routes containing private AS numbers. Unless they come from allowed customers.
    - Routes that do not start with peer's AS number, unless routes come from a route server (out of the scope of this case study).
    - Routes from customers that do not contain AS numbers belonging to the given customer or for what this customer is authorized to transit to.
        - Worse, but valid solution is to avoid accepting as-paths longer than one. This is valid unless the customer is authorized to provide transit to certain destinations.
    - Routes that contain its' own AS number coming external peers. This overrides BGP normal behavior and must be forcefully configured. The RFC warns that the impact may be severe.
- To avoid advertisements:
    - With non-empty as-path. Unless the network provides transit for these prefixes.
    - With upstream AS numbers in the as-path to their peering ASes unless they are willing to provide transit.

- With private AS numbers in the as-path.

The patterns are defined by regular expressions and they can be tested through the show interface as follows. The regex language is defined at 11.17 of Frrouting manual. It's exactly the same than Quagga.

Preventing the route leak explained before

To show the performance of this protection, we repeat the route leak explained before.

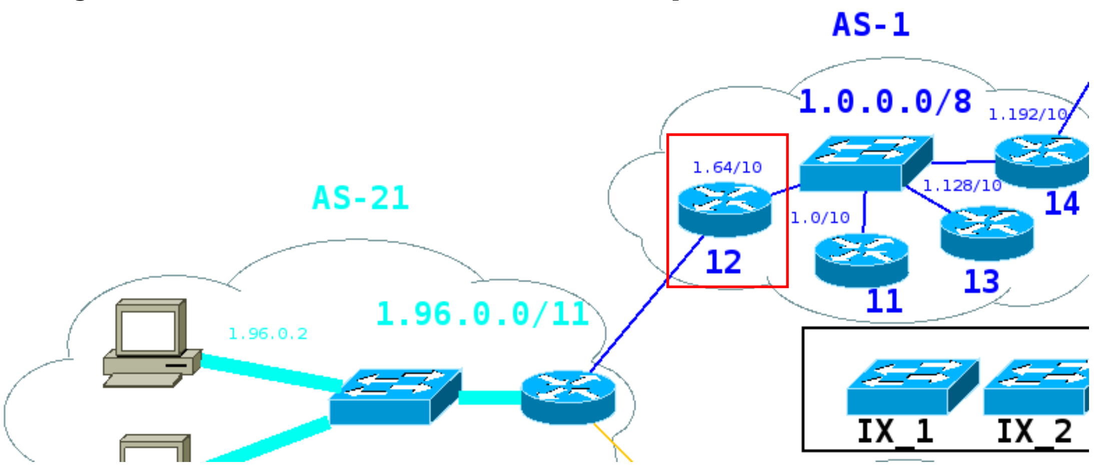
As seen before, as1card2 selects to transit traffic through as21card1 instead of doing through the normal paths.

We defeated the route leak case by filtering coming from Tier 2 AS routers to Tier 1 AS routers that do not just match with the as-path "21".

With that in mind, a new environment was deployed, and the route leak attempt was repeated again without success from as21card1.

Next-Hop Filtering

The most common way to publish a route is to set the next hop to the router that makes the advertisement. This is commensurate, since usually the advertisement receivers would not be able to reach the router that truly offers access to the given prefix but through the advertiser. To be clearer, we will explain this over the environment:



Let's say that as1card2 advertises a route to 1.192.0.0/10 with next-hop 172.16.11.4 to AS21. That makes no sense to do it so, since as21card1 has not a way to reach that host on AS1 private's network. So, as1card2 makes the advertisement to this network replacing next-hop by itself on a network range visible by as21card1.

This is done on the advertiser side with the configuration directive:

```
neighbor 10.11.21.1 next-hop-self
```

The protocol and implementations of BGPv4 allow to replace and change this behavior to support different setups. For example, at IXPs where routers just receive routes from a route server. Route servers will never want to set up next-hop-self, but they want to instruct routers to use certain next hops to reach certain networks.

This functionality allows an attacker to redirect traffic through another hop, therefore, it should be filtered and overridden with the peer address on the side that receives the advertisement. Unless working in a route server setup.

This is filtered at the route receptor side with the following configuration directives:

Route maps allow to both filter and apply actions to received routes. Not only for BGP but for all routing protocols offered by Frrouting or Quagga suites.

This sets up a next-hop overwrite with the peer-address when a route is received.

```
route-map AntiSpoofNextHop permit 10
    match ip next-hop peer-address
route-map ReplaceNextHop permit 10
    set ip next-hop peer-address
```

After that, the route-map must be applied to the peer, so every next-hop received on route advertisements from this peer gets replaced with the peer-address or checked and discarded if it does not match the peer-address:

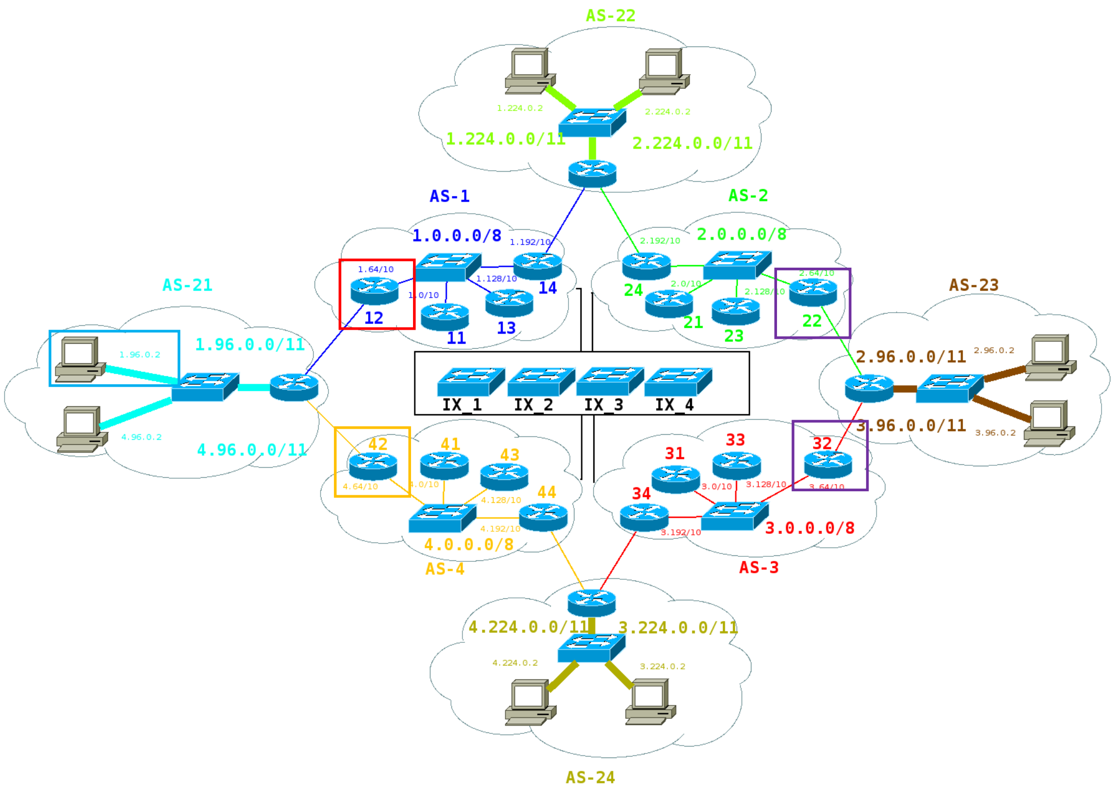
```
address-family ipv4 unicast
…
    neighbor 10.11.21.1 route-map [AntiSpoofNextHop|ReplaceNextHop] in
…
exit-address-family
```

Attack: Next-Hop Spoof

When testing TCPMD5 Signature, we performed a MITM attack over an unsecured BGPv4 – TCP session and next-hop got replaced by another valid next-hop on the given network segment. We demonstrated that it possible to make the victim to transit more traffic than expected to a given destination.

The objective is the same, to make the victim to move more traffic than expected.

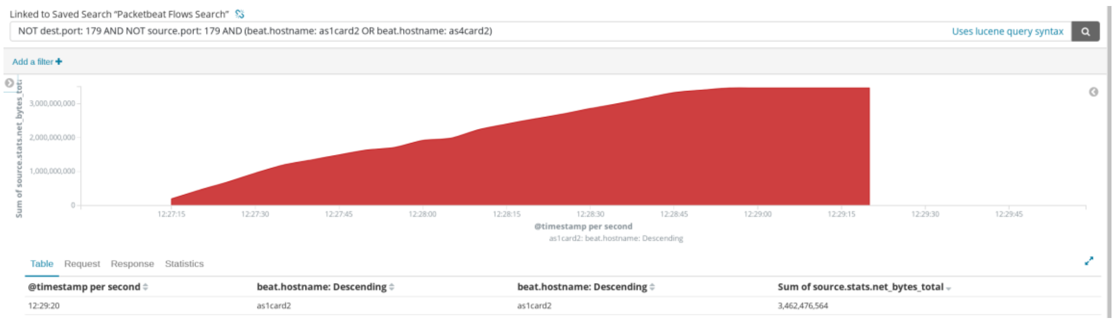
The victim on this case is as4card2. The attack takes place at IX_2 and the attacker is as1card2 that wants to make as4card2 to transit more traffic than he expects to. Regarding environment's network topology, both victim and attacker are connected at IX_2.



To show the normal behavior we will measure bandwidth seen at both victim and attacker in a normal case, when all clients are generating traffic to client_21_1_96_2.

We use Iperf to make all clients in the network out of AS21 to generate traffic to client_21_1_96_2, located inside AS21 networks. The bidirectional data amount to be transmitted is 100Mb from each generator and direction.

Once executed, it can be appreciated that as1card2 is transiting the whole traffic from AS23 to AS21, for client_21_1_96_2.


Linked to Saved Search "Packetbeat Flows Search"

NOT dest.port: 179 AND NOT source.port: 179 AND (beat.hostname: as1card2 OR beat.hostname: as4card2)

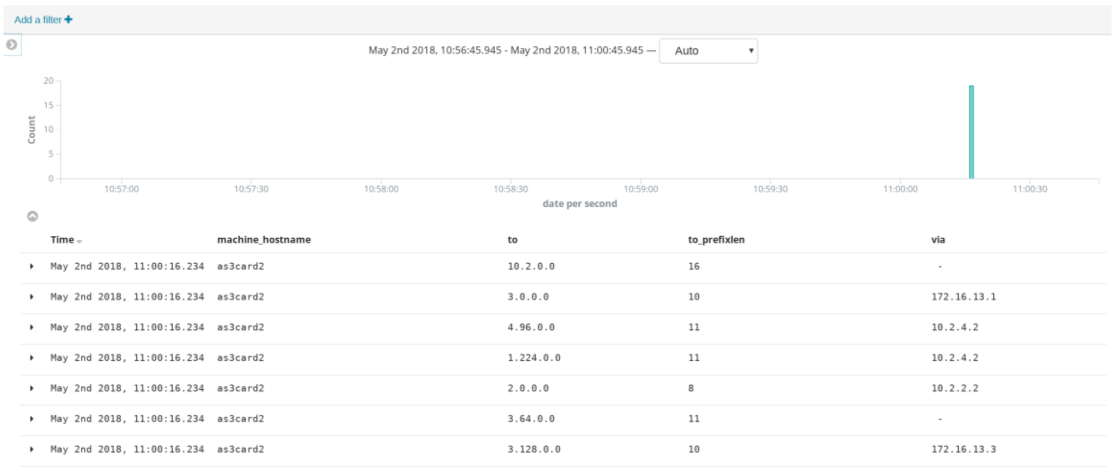
@timestamp per second	beat.hostname: Descending	beat.hostname: Descending	Sum of source.stats.net_bytes_total
12:29:20	as1card2	as1card2	3,462,476,564

Before the attack is performed, the routing table of as3card2 and as2card2 returns that client is reached through as1card2.

```
root@as3card2:~# ip route get 1.96.0.2
1.96.0.2 via 10.2.1.2 dev IX_2 src 10.2.3.2 uid 0
    cache
```

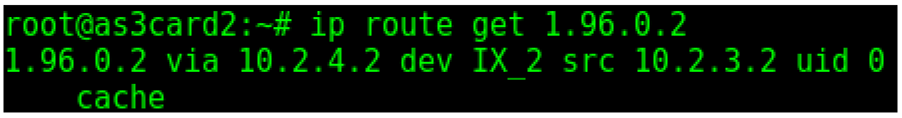
Now, as1card2 decides that this is not fair and changes its' advertisement to network 1.96.0.0/11 to set next-hop through as4card2. We modify the announcement to both as2card2 and as3card2, so advertisements are made with next hop pointing to the victim, as4card2. We use a route-map to change these advertisements from as1card2.

That produced 19 route updates on as3card2 and as2card2 kernel's routing tables:



Time	machine_hostname	to	to_prefixlen	via
May 2nd 2018, 11:00:16.234	as3card2	10.2.0.0	16	-
May 2nd 2018, 11:00:16.234	as3card2	3.0.0.0	10	172.16.13.1
May 2nd 2018, 11:00:16.234	as3card2	4.96.0.0	11	10.2.4.2
May 2nd 2018, 11:00:16.234	as3card2	1.224.0.0	11	10.2.4.2
May 2nd 2018, 11:00:16.234	as3card2	2.0.0.0	8	10.2.2.2
May 2nd 2018, 11:00:16.234	as3card2	3.64.0.0	11	-
May 2nd 2018, 11:00:16.234	as3card2	3.128.0.0	10	172.16.13.3

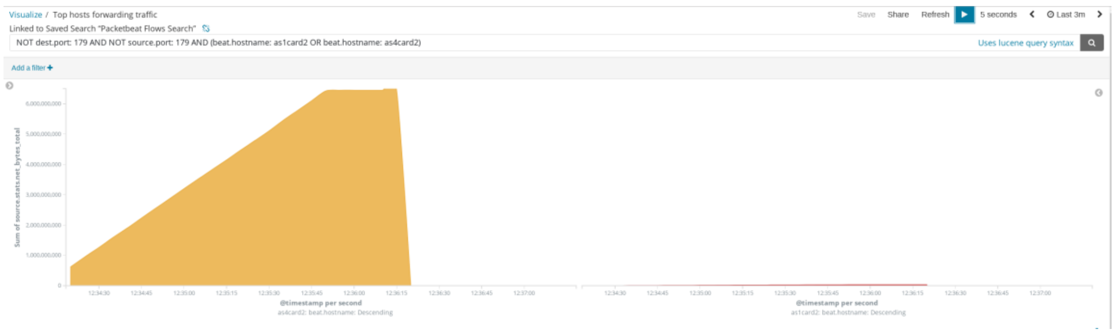
Now as2card2 and as3card2 reach the client through as4card2, the victim.



```
root@as3card2:~# ip route get 1.96.0.2
1.96.0.2 via 10.2.4.2 dev IX_2 src 10.2.3.2 uid 0
    cache
```

The same steps are applied to as2card2. This way all the traffic coming from the other side of the network, AS23 is routed through as4card2.

Now, the traffic transits through as4card2 and therefore, that offloads as1card2 as it wanted to. Not the whole traffic, since it's a bidirectional test and TCP ACKs from the client are getting routed through as1card2, back to the clients of this Iperf server.



As shown, as1card2 can make other routers to send more traffic to the victim, as4card2. This technique can be used in several ways to attack or just to generate revenue by influencing routers out of the control of the victim to drive more traffic through the victim and for example, generate revenue.

5.3.5. Attacks over the whole network

The purpose of this section is to group attacks that are not mitigated through just one of the previously explained security measures. They may involve two or more defenses, and, in some cases, the attack will only be mitigated but not impossible.

While defenses explained previously, mitigate or make impossible the exercise of certain types of attacks, there are others that are more complicated or nearly impossible to defeat. On the current environment the following defenses are implemented:

- Linux Kernel and Firewall have been set up to avoid routing traffic to private CIDR network domains. In addition, kernel's behavior has been changed to mitigate DoS or DDoS attacks over routers. Considering limitations in term of RAM and processor, attacks that involve high loads of traffic to reset a router or a BGPv4 session are not considered.
- TTL-Security with maximum hop distance of 1. That makes impossible to perform attacks from outside of BGPv4 speaker's network segments.
- TCP-MD5 Signature. The password is assumed to be strong. Therefore, an attempt to crack it would take much longer than in the test performed. It can be assumed that MITM attacks are not possible.
- Prefix Filters have been applied to both avoid leaking internal ASes structure and to show the performance of route leak attack performed by a multi-homed customer.
- BGP Route flap dampening. Dampening on flapping routes is enabled as exposed on (RFC7454 - BGP Operations and Security).
- Maximum prefixes on a peering have been limited and limited routers memory is protected.
- AS path Filters have been enabled and customers, AS21, AS22, AS23 and AS24 are not allowed to advertise routes longer than one and that do not contain exactly it's AS number.
- Next Hop Filters are enabled, and Next Hop spoofing is prevented.

Before explaining the attacks it's mandatory to regard the following basis about IP routing and BGPv4.

There are two algorithms that manage respectively the BGPv4 route selection and the routing itself.

The BGPv4 Route selection algorithm manages what routes go in and out from the effective routing/forwarding table (AKA the kernel routing table). The behavior may vary between manufacturers and the behavior can be changed by network administrators through management interfaces. The most common case is that it chooses as the best route, to a given prefix, the shortest path in terms of autonomous systems.

Routers apply the Longest Prefix Algorithm (LPM) for making the decision about the next hop and the interface each IP packet should be sent onto. Implemented in the kernel, it looks up each IP packet's destination IP address into one or more forwarding/routing tables and computes the best match, the LPM. This algorithm is the basis of IP routing and its behavior cannot be modified without tampering the forwarding table implementation.

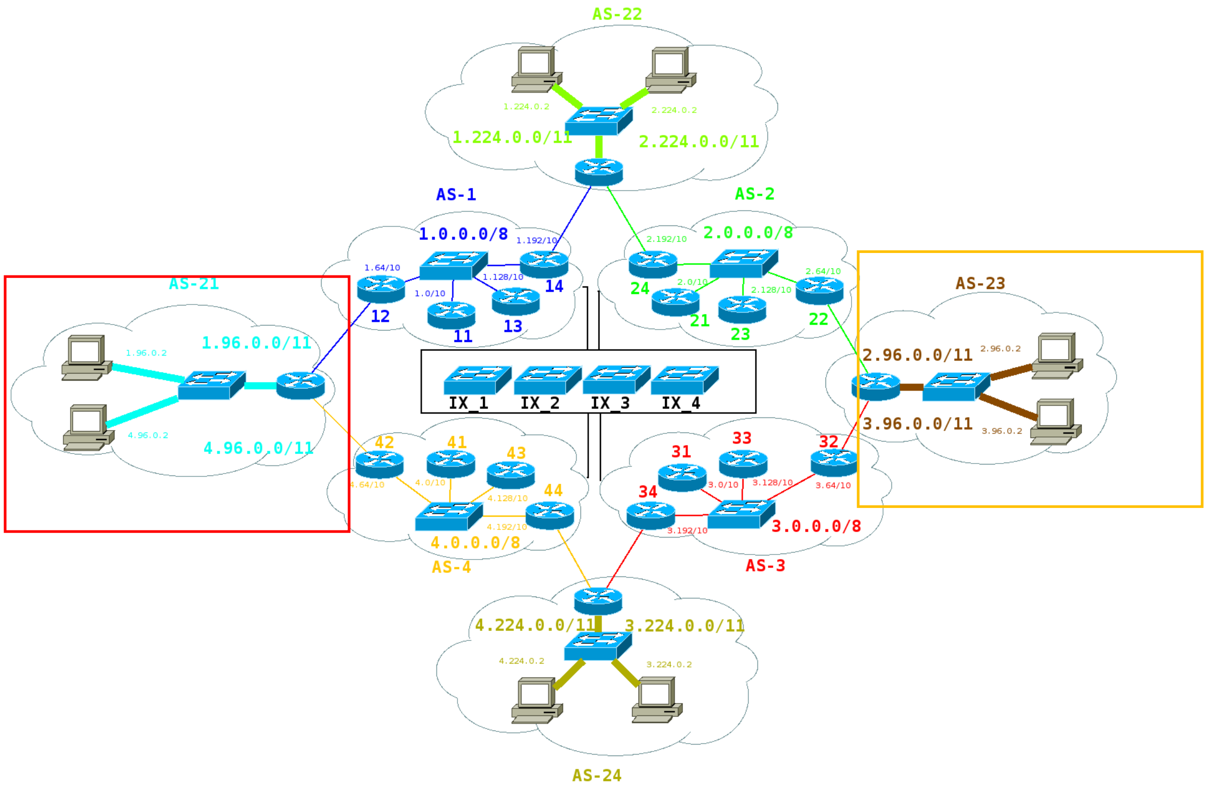
Therefore, exhaustive filters should be applied before updating these tables with routes received from outside, since they will change the router behavior.

Prefix and Sub-prefix hijacks

The Prefix and Sub-prefix Hijacks are some of the worst attacks that can be exercised against a network of BGPv4 routers.

Both require knowledge about BGPv4 route selection algorithm and LPM.

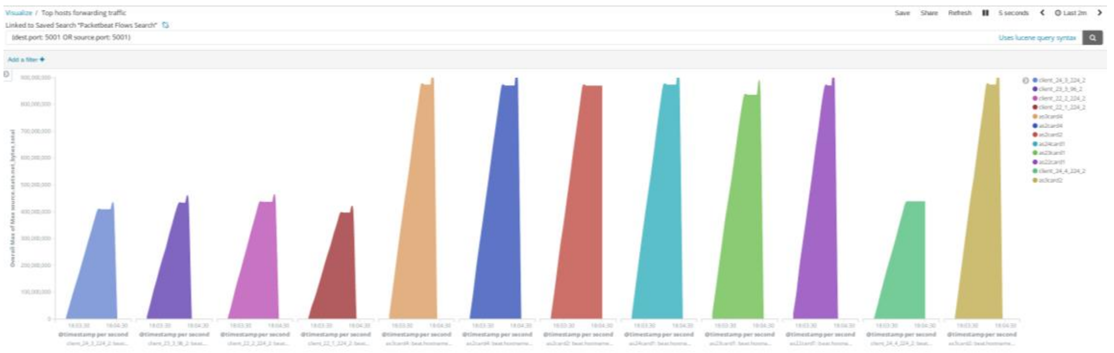
Over the case study, the AS21 wants to hijack traffic going to and from the victim 3.96.0.2, client_23_3_96_2:

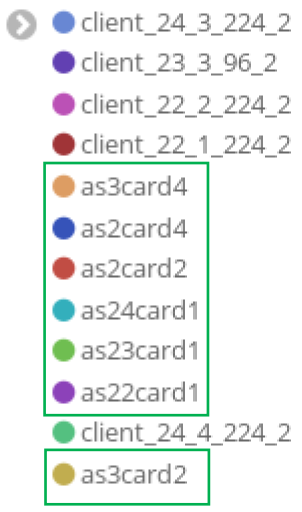


To do it, the attacker may go straight forward and advertise both 3.96.0.0/11 and 2.96.0.0/11 prefixes.

To simulate that, we configured both down_1:1 and down_4:1 virtual interfaces on as21card1 to have the same addresses of AS23 clients.

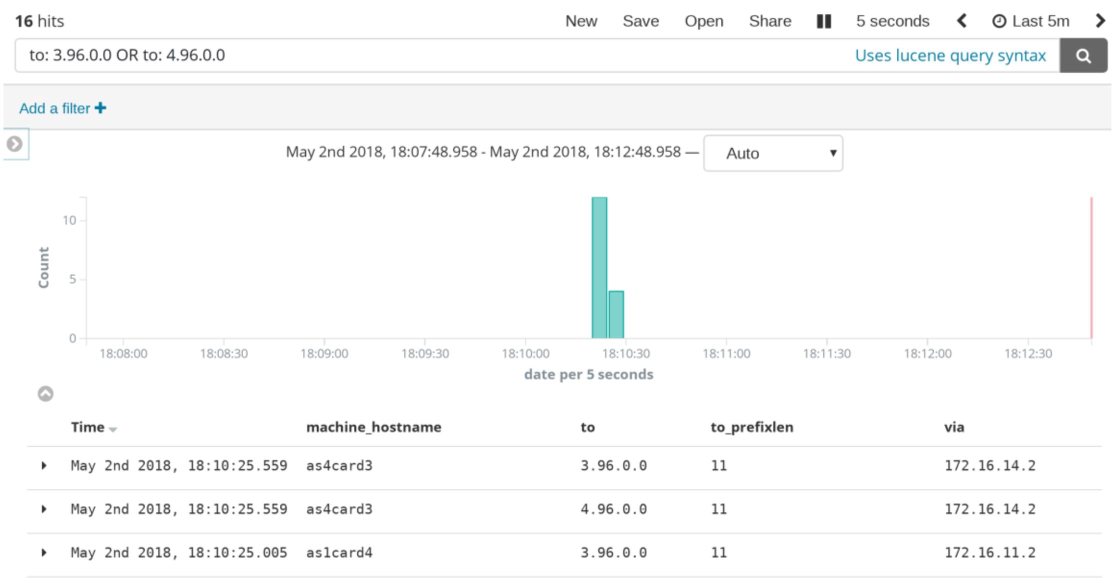
Before the attack is executed, clients from AS22 and AS24 generate traffic to AS23. The traffic volume measured on the overall is as follows:





The routers forwarding traffic were as3card4, as2card4, as2card2, as24card1, as23card1, as22card1 and as3card2.

Now the starts the attack. The effect was obviously spread over kernel routing tables of several routers:


16 hits

to: 3.96.0.0 OR to: 4.96.0.0

May 2nd 2018, 18:07:48.958 - May 2nd 2018, 18:12:48.958 — Auto

	Time	machine_hostname	to	to_prefixlen	via
▸	May 2nd 2018, 18:10:25.559	as4card3	3.96.0.0	11	172.16.14.2
▸	May 2nd 2018, 18:10:25.559	as4card3	4.96.0.0	11	172.16.14.2
▸	May 2nd 2018, 18:10:25.005	as1card4	3.96.0.0	11	172.16.11.2

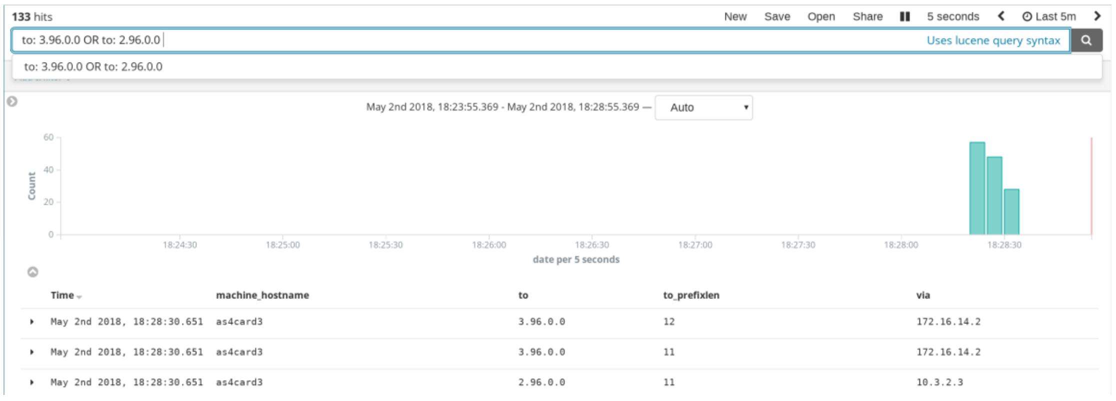
8 routers got updates on them kernel routing tables with destination [3|4].96.0.0/11. They were as4card3, as1card4, as1card2, as4card1, as1card1, as1card3, as4card2, as4card4. But none of them were customer ASes so announcing these prefixes will only

allow the attacking AS to hook a portion of the whole traffic AS23 would receive. This portion is the traffic that travels through this list of routers. In the previous case, none.

From this point, the attacker, AS21, decides to change the advertised routes to [3|4].96.0.0/12. Therefore, changing the attack from a prefix hijack to a sub-prefix hijack.

```
address-family ipv4 unicast
…
        network 2.96.0.0/12
        network 3.96.0.0/12
…
exit-address-family
```

Once done, it produces 133 updates on kernel routing tables on the overall of the network:

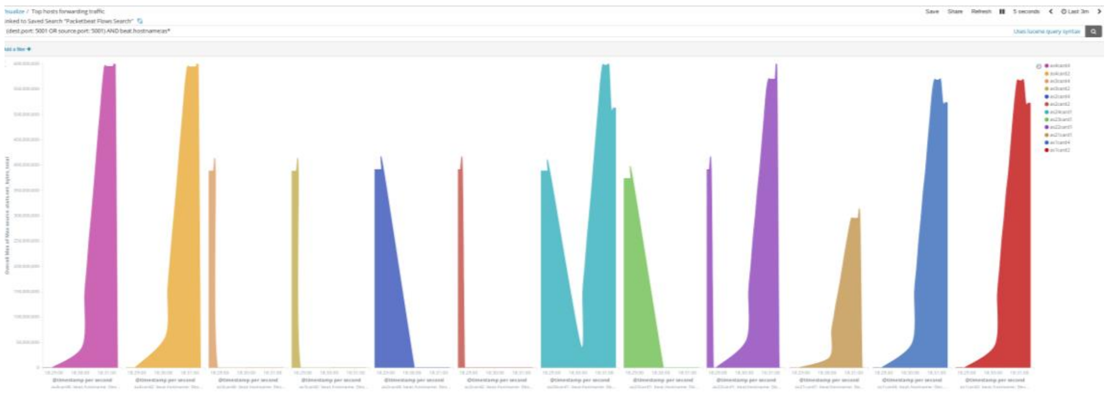

133 hits

to: 3.96.0.0 OR to: 2.96.0.0

May 2nd 2018, 18:23:55.369 - May 2nd 2018, 18:28:55.369 — Auto

```
Time                          machine_hostname   to         to_prefixlen   via
May 2nd 2018, 18:28:30.651    as4card3           3.96.0.0   12             172.16.14.2
May 2nd 2018, 18:28:30.651    as4card3           3.96.0.0   11             172.16.14.2
May 2nd 2018, 18:28:30.651    as4card3           2.96.0.0   11             10.3.2.3
```

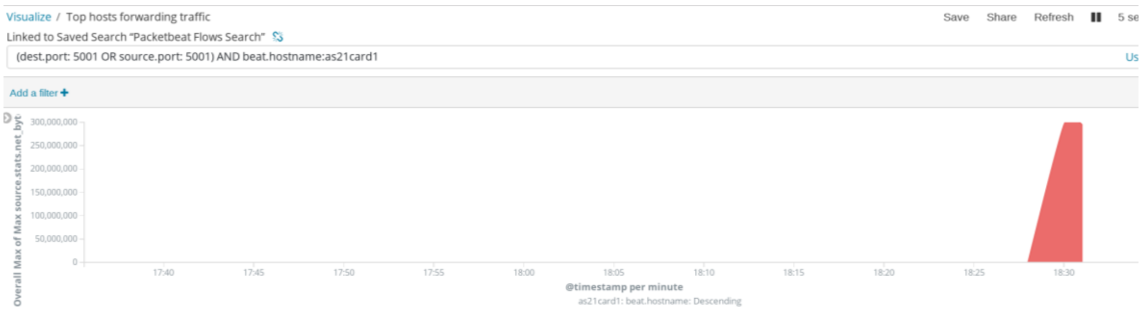
After repeating the traffic generation test, the change can be appreciated:



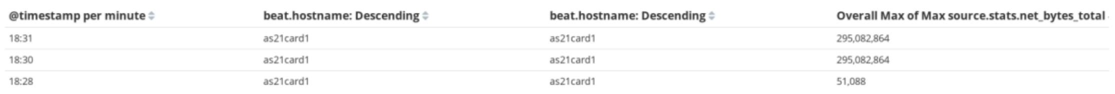
New boys have joined the party:

● as3card4
● as3card2
● as2card4
● as2card2
● as24card1
● as23card1
● as22card1
● as4card4
● as4card2
● as21card1
● as1card4
● as1card2

And as follows, AS21 who did not receive traffic before, has started to receive it. The following histogram shows the traffic volume measured at AS21 since this attack was started:

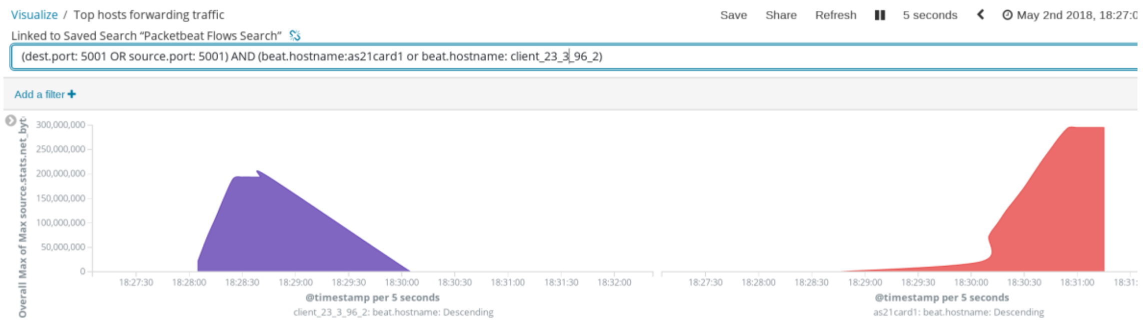

Visualize / Top hosts forwarding traffic
Linked to Saved Search "Packetbeat Flows Search"
(dest.port: 5001 OR source.port: 5001) AND beat.hostname:as21card1

Concretely, it started receiving some traffic at 18:28 when the advertisement was made.



@timestamp per minute	beat.hostname: Descending	beat.hostname: Descending	Overall Max of Max source.stats.net_bytes_total
18:31	as21card1	as21card1	295,082,864
18:30	as21card1	as21card1	295,082,864
18:28	as21card1	as21card1	51,088

In addition, the following histogram shows how the legitimate client stops receiving traffic in favor of the attacker:


Visualize / Top hosts forwarding traffic
Linked to Saved Search "Packetbeat Flows Search"
(dest.port: 5001 OR source.port: 5001) AND (beat.hostname:as21card1 or beat.hostname: client_23_3_96_2)

● client_23_3_96_2
● as21card1

5.4. Protections Analyses – Phase 2

This section contains the defenses analyzed inside the scope of Secure Inter-Domain Routing.

The environment used is exactly the environment defined at (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2) and it inherits all deployed protections from the previous phase.

5.4.1. Objectives

The main objective is to illustrate the security improvement provided by protections and configurations explained at (RFC7454 - BGP Operations and Security) on section SIDR - Secure Inter-Domain Routing, in front of some attacks.

While doing it, the automated generation of this environment is improved to integrate explained protections. For each security measure a new environment is generated. This way the security of the whole network is improved step by step.

The section before, showed the performance of defenses applied on BGPv4 speakers, sessions and routing. This one, extends the latter to show the benefits of implementing RPKI, part of SIDR Infrastructure, against Prefix and Sub-prefix Hijack attack mainly, while introducing new ones.

5.4.2. Defeating Prefix and Sub-prefix hijacks through RPKI

This section gives a brief introduction to RPKI, explains the configuration applied over the environment and analyzes the performance against Prefix and Sub-prefix hijack attacks.

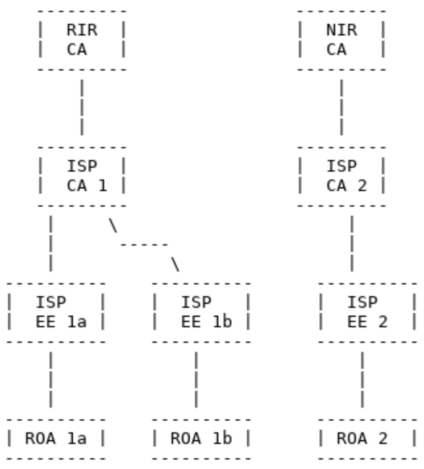
RPKI Basis

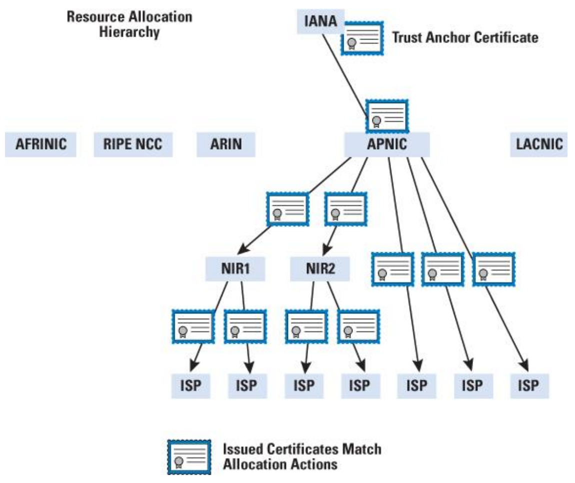
Resource Public Key Infrastructure is a part of SIDR (RFC6480 - An Infrastructure to Support Secure Internet Routing). RPKI defines the application of public key infrastructure to the arborescent structure of IP networks. But it's bigger than that. The following standards define critical parts such as X509 certificate extensions or ROAs' structure (note that only the most important ones are shown):

- (RFC6841 - A Profile for Resource Certificate Repository Structure),
- (RFC6842 - A Profile for Route Origin Authorizations (ROAs))
- (RFC6487 - A Profile for X.509 PKIX Resource Certificates)
- (RFC6488 - Signed Object Template for the Resource Public Key Infrastructure (RPKI))
- (RFC6493 - The Resource Public Key Infrastructure (RPKI) Ghostbusters Record)
- (RFC6810 - The Resource Public Key Infrastructure (RPKI) to Router Protocol)
- (RFC6811 - BGP Prefix Origin Validation)

Certificates are commonly used to identify a subject, but this is not the case here, certificates are referred as Resource Certificates and they attest the allocation by the issuer of IP addresses or AS numbers to the subject.

IANA is defined as the ROOT Certification Authority, RIRs as subordinate CAs, NIRs/LIRs/ISPs as subordinates of the previous ones and so on. The pictures below, from (RFC6480 - An Infrastructure to Support Secure Internet Routing) and (Cisco®), show the concrete application:





In a nutshell we can say that each prefix – asn pair has its' own certificate and it's signed by the CA that owns this prefix, the issuer.

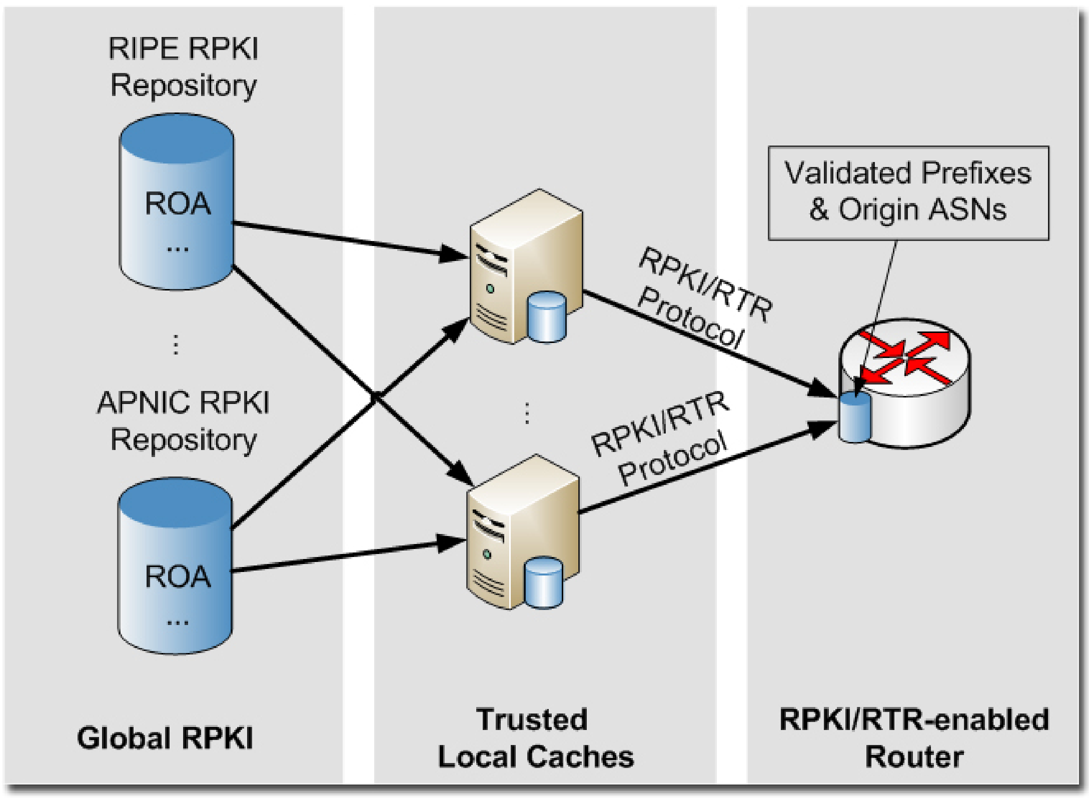
The trust anchor, the CAs that some network operator trusts, are ISPs, LIRs, NIRs, RIRs and IANA. They can sign and update repositories with signed resource certificates to authorize a third party ASN to advertise a certain prefix inside them networks.

But the question now is:

How do we take this information to the routers, so they can validate against the RPKI repository ?

Obviously, make the routers to perform cryptographic validations and to have the full repository stored inside is not currently affordable, considering the size of the Internet. Because of that, the certificates are stored in repositories. Each network operator should clone and keep up to date them. This is done by Validators, that perform the hard work of checking resource certificates' signatures, pulled from the repositories. Now, a new element arises to answer the question: the intermediate caches.

These intermediate caches speak both with local or remote repository clones and routers.



These intermediate caches store the validator output to offer them to the routers. Therefore, the cache stores Validated ROA Payloads, VRPs.

Practical application – Defeating Prefix and Sub-Prefix Hijacks

As explained at (Foces Vivancos, Securing The BGPv4: PEC3 Security Analyses - Part 2) 2.3.2 the RPKI-RTR speaker, the cache, receives a JSON file defining the routing policy and this way, it acts as trusted cache for routers. Regarding environment definition, the cache server is connected to routers through the management network and RPKI/RTR travels in plain text through this assumed secure network.

To accomplish defined objectives and show the performance of this security measure in front of Prefix and Sub-prefix hijacks, all routers have been configured to check advertisements and act as stated on (RFC7454 - BGP Operations and Security):

- *If a corresponding ROA (Route Origin Authorization) is found and is valid, then the prefix SHOULD be accepted.*
- *If the ROA is found and is INVALID, then the prefix SHOULD be discarded.*
- *If a ROA is not found, then the prefix SHOULD be accepted, but the corresponding route SHOULD be given a low preference.*

That is a very smart tactic, because it discards invalid advertisements while keeping legacy operation untouched.

We use route maps to configure routers conformant with the standard.

These route maps refer to the following result of the validation of a prefix – asn pair against the RPKI/RTR Table: invalid, not found or valid. As we can find on (RFC6810 - The Resource Public Key Infrastructure (RPKI) to Router Protocol), these prefix origin validation results are assigned in the following cases:

- NotFound: No VRP(Validated ROA Prefix) Covers the Route Prefix.
- Valid: At least one VRP Matches the Route Prefix.
- Invalid: At least one VRP Covers the Route Prefix, but no VRP Matches it.

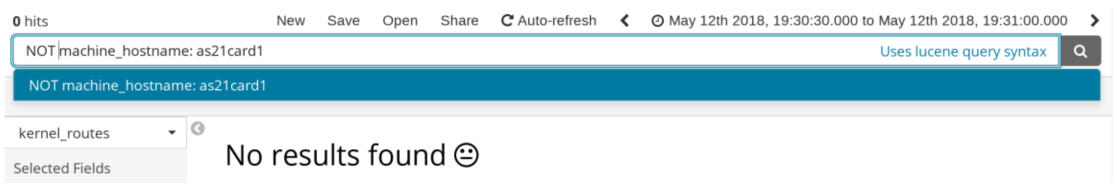
As the standard defines:

Covered: A Route Prefix is said to be Covered by a VRP when the VRP prefix length is less than or equal to the Route prefix length, and the VRP prefix address and the Route prefix address are identical for all bits specified by the VRP prefix length. (That is, the Route prefix is either identical to the VRP prefix or more specific than the VRP prefix.)

Matched: A Route Prefix is said to be Matched by a VRP when the Route Prefix is covered by that VRP, the Route prefix length is less than or equal to the VRP maximum length, and the Route Origin ASN is equal to the VRP ASN.

Re-executing Prefix and Sub-prefix hijacks

After the environment is deployed. The steps to perform the prefix a sub-prefix hijacks are repeated. There were no changes on the kernel routing tables. All routers discarded the advertisements, since the AS21 is not authorized by the ROAs to advertise these prefixes:



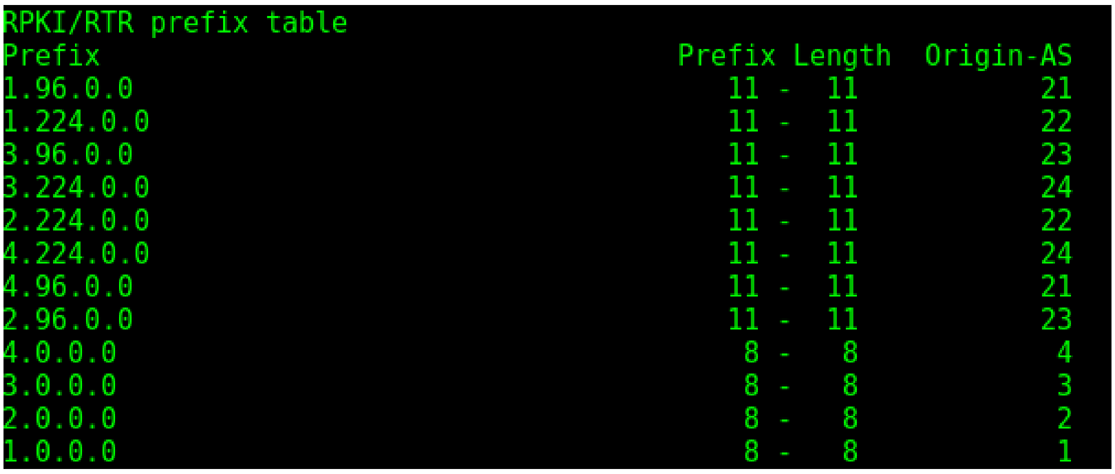
NOT machine_hostname: as21card1

NOT machine_hostname: as21card1

No results found 😐

Obviously, AS21Card1 had updates on its routing table due to the declaration of the virtual interfaces and IP addresses.

The RPKI/RTR prefix table on the routers contain the values shown below. Therefore, the rpki validation returns invalid for the advertisements made by AS21:


```
RPKI/RTR prefix table
Prefix                                  Prefix Length  Origin-AS
1.96.0.0                                   11 -  11           21
1.224.0.0                                  11 -  11           22
3.96.0.0                                   11 -  11           23
3.224.0.0                                  11 -  11           24
2.224.0.0                                  11 -  11           22
4.224.0.0                                  11 -  11           24
4.96.0.0                                   11 -  11           21
2.96.0.0                                   11 -  11           23
4.0.0.0                                     8 -   8            4
3.0.0.0                                     8 -   8            3
2.0.0.0                                     8 -   8            2
1.0.0.0                                     8 -   8            1
```

Combining that with the call to the route map upon receiving advertisements, routers are able to decide that the routes are not valid and therefore, drop them.

5.4.3. Next Hop Hijack

Both prefix and sub-prefix hijacks are not possible if RPKI is deployed and routers are working with it. But, that does not provide protection in terms of the AS path. The only restriction it provides is that the last AS on the path is the owner of the advertised prefix.

Therefore, the attacker can advertise any route that ends on the victim AS.

Over the current simulation, AS21 keeps willing to hijack traffic from AS23. To simulate that, let's imagine that AS21 offers internet service to his clients and therefore, AS1 and AS4 cannot filter advertisements with more than one AS on the path made by AS21.

Remember that we had protected Tier1 routers from leaking internal structure through an AS-path filter. This is now disabled.

To illustrate the effect of the Validated ROA Payloads, stored in the cache, two cases of NextHop Hijack are analyzed. On the first one, we misconfigure the maxLength attribute of each ROA on the cache server by increasing the maxLength attribute to 32. On the second one it takes the prefix length.

Wide Open Max Prefix Length VRPs

The maxLength attribute of cached VRPs has now the value of 32.

Over this context, the attacker AS21 has to change its configuration to allow advertisements for networks 2.96.0.0/1[1|2] and 3.96.0.0/1[1|2]. After that, it defines a route-map to add AS23 to advertisements made for certain networks (2.96.0.0/11 and 3.96.0.0/11).

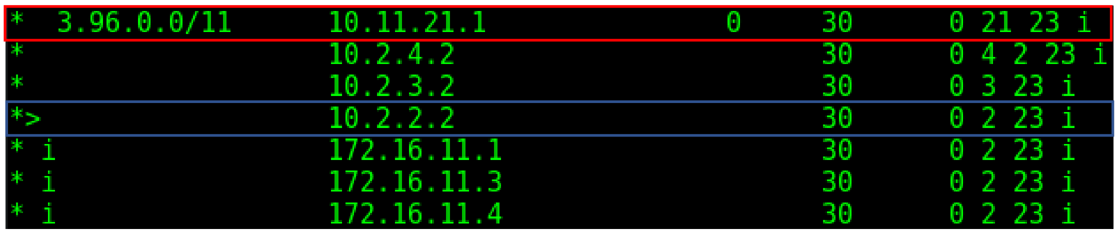
It configures the interfaces as it did previously, on the Prefix and Sub-Prefix hijack.

To finish, it just declares the bogus route announcements, since these are not valid in the AS graph.

For this case, we try to perform NextHop hijack for the whole prefixes advertised by AS23.

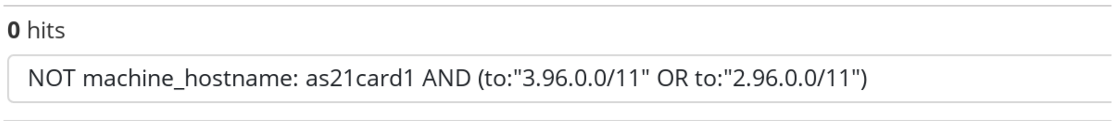
That produces route advertisements to have the following as-path “21 23”.

The impact: the routing table of As1Card2 has a new entry:



```
*  3.96.0.0/11       10.11.21.1               0      30         0 21 23 i
*                    10.2.4.2                        30         0 4 2 23 i
*                    10.2.3.2                        30         0 3 23 i
*>                   10.2.2.2                        30         0 2 23 i
* i                  172.16.11.1                     30         0 2 23 i
* i                  172.16.11.3                     30         0 2 23 i
* i                  172.16.11.4                     30         0 2 23 i
```

The BGP route selection algorithm is not selecting these routes, since it has already another way to reach the given prefixes. The paths are of the same length in terms of AS-Path. Therefore, it keeps the route through 10.2.2.2 through AS path 2 23. Therefore, there is no impact on the effective forwarding table (the kernel routing table) on any router on the network:



0 hits

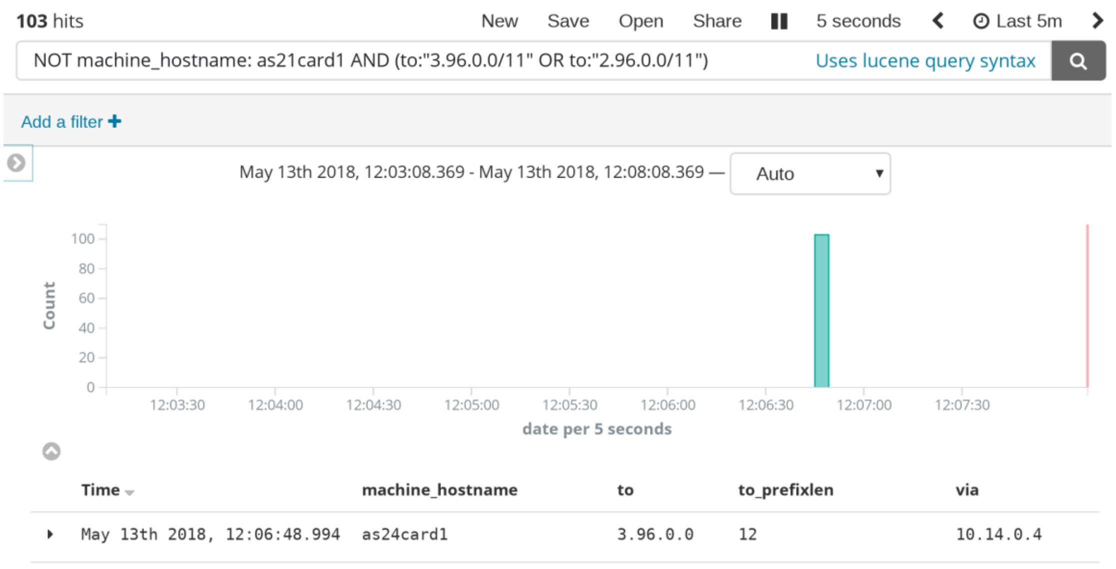
NOT machine_hostname: as21card1 AND (to:"3.96.0.0/11" OR to:"2.96.0.0/11")

It’s important to regard that the routes are accepted since they are valid in RPKI terms, but they are not used since others of the same length are currently in use. The attacker may try to reset BGPv4 sessions of this router with others until his route is selected. Therefore, that demonstrates that every protection means.

On this context, the traffic volume measurement is not applicable, since there are no changes on effective routing tables.

With that in mind, let’s trick it by both the BGP route selection algorithm and Longest Prefix Matching, by advertising the prefixes 3.96.0.0/12 and 2.96.0.0/12:

Once done it produces 103 kernel routing table updates on the whole network.


103 hits

New Save Open Share 5 seconds Last 5m

NOT machine_hostname: as21card1 AND (to:"3.96.0.0/11" OR to:"2.96.0.0/11")

Uses lucene query syntax

Add a filter

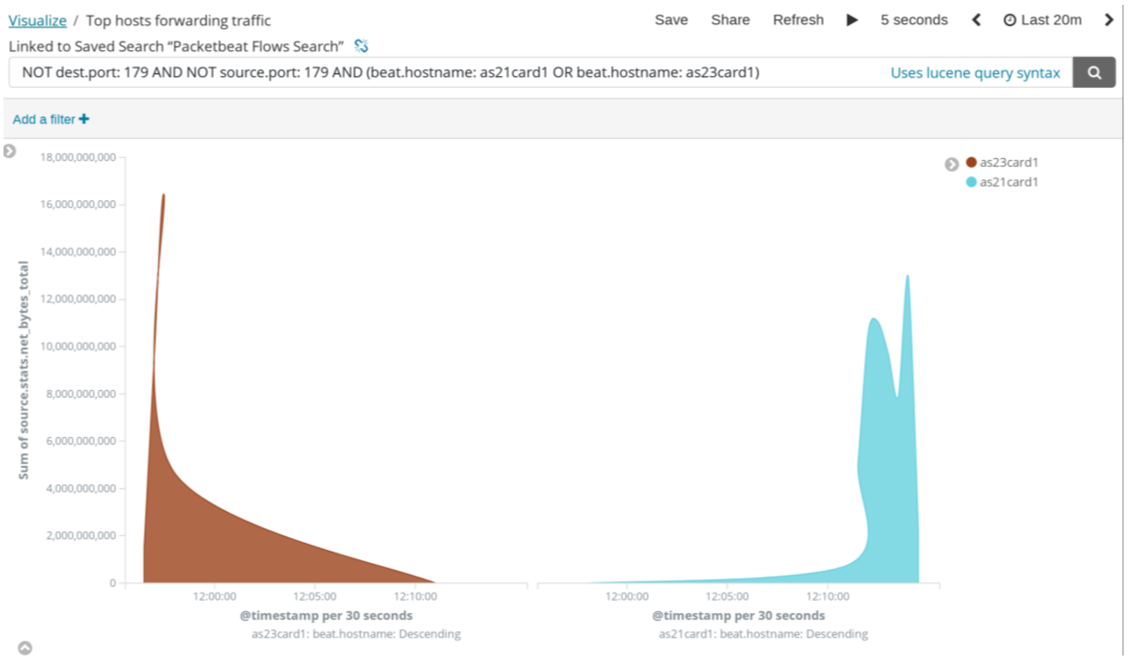
May 13th 2018, 12:03:08.369 - May 13th 2018, 12:08:08.369 — Auto

Time	machine_hostname	to	to_prefixlen	via
May 13th 2018, 12:06:48.994	as24card1	3.96.0.0	12	10.14.0.4

These advertisements are accepted and effectively used, because:

- The advertisements are accepted because of the value (32) of the maxLength attribute of VRPs in the cache server. This allows advertisements of prefixes from length 11 to 32 from AS23.
- Once accepted, the advertised routes are introduced in the kernel routing tables and then, the Longest Prefix Matching makes the rest.

The histogram below shows that the attacker has successfully hijacked AS23's traffic:


Visualize / Top hosts forwarding traffic

Linked to Saved Search "Packetbeat Flows Search"

Save Share Refresh 5 seconds Last 20m

NOT dest.port: 179 AND NOT source.port: 179 AND (beat.hostname: as21card1 OR beat.hostname: as23card1)

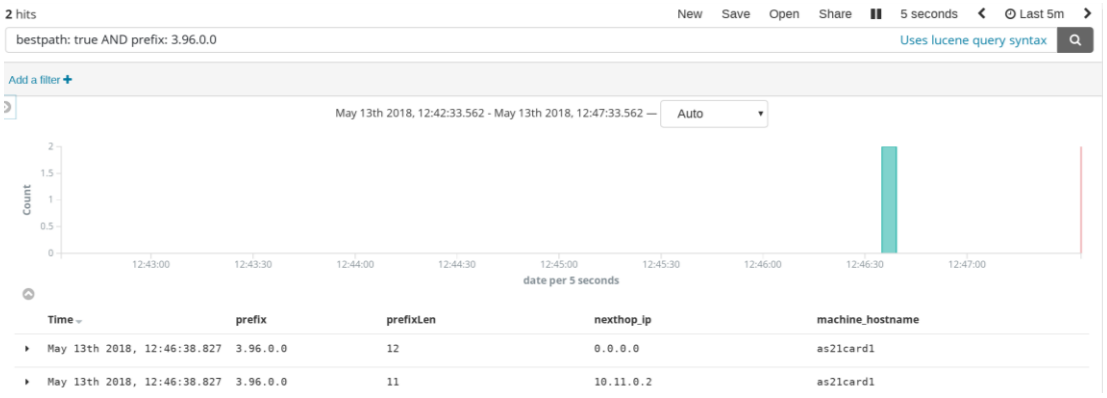
Uses lucene query syntax

Add a filter

Short Max Prefix Length VRPs

After that, we roll back the configuration to define valid ROAs for prefixes with exact lengths, for this case, maxLength attribute takes the value of 11.

This way, the sub-prefixes advertisements, 3.96.0.0/12 and 2.96.0.0/12 are not valid and they are discarded by all routers in the network:



2 hits

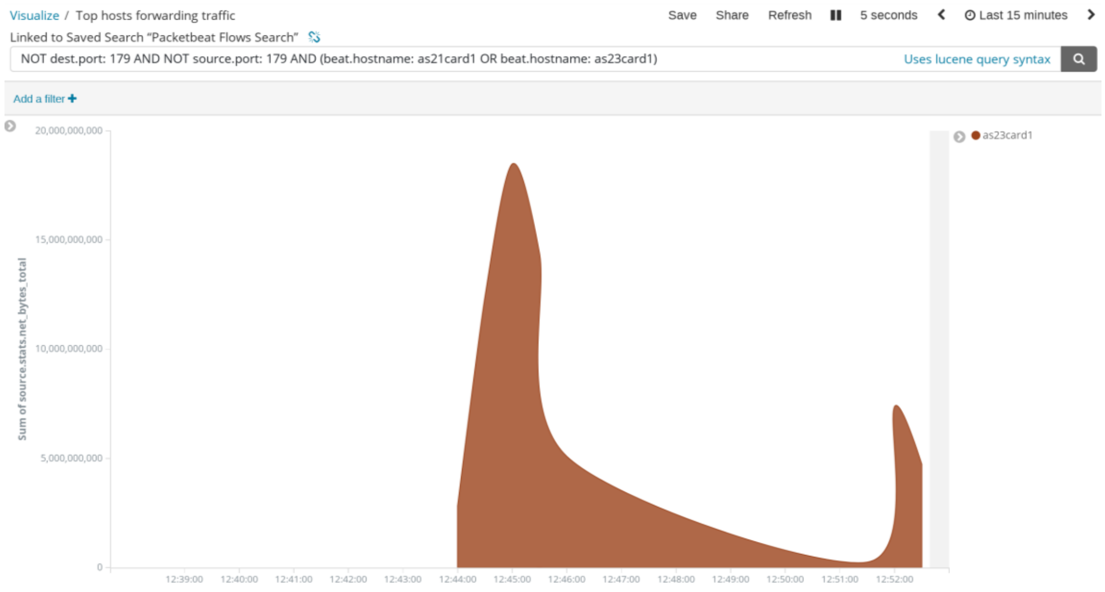
bestpath: true AND prefix: 3.96.0.0

May 13th 2018, 12:42:33.562 - May 13th 2018, 12:47:33.562

Time	prefix	prefixLen	nexthop_ip	machine_hostname
May 13th 2018, 12:46:38.827	3.96.0.0	12	0.0.0.0	as21card1
May 13th 2018, 12:46:38.827	3.96.0.0	11	10.11.0.2	as21card1

Note that the only one who has changes on the routing table is As21Card1, the attacker.

The attack is not successful when the ROAs are correctly defined:



Linked to Saved Search "Packetbeat Flows Search"

NOT dest.port: 179 AND NOT source.port: 179 AND (beat.hostname: as21card1 OR beat.hostname: as23card1)

On this point the attacker can only attempt to perform a NextHop hijack against valid prefixes on the ROAs (VRPs) otherwise routes will be dropped.

The attack will be successful when the BGP route selection algorithm selects the bogus routes. This will happen only when the attacker announces the shortest path the victim router knows to the legitimate AS.

6. Conclusions

As stated before, the project has gone through 3 main phases: State of art, environment design and deployment, protection analyses loop.

From the **first phase**, State of art, we conclude that BGPv4 is vulnerable by design and that protections are out of the scope of the protocol definition. The protections to make devices to exchange routes in a secure manner are provided by elements surrounding BGPv4 speakers' operation.

This phase shows that it's hard to convince all entities connected in the internet to standardize certain technologies or to implement protections that won't provide benefits to them. This is the purpose of Mutually Agreed Norms for Routing Security, a global initiative to improve the security of Internet's routing system.

While not formally documented in this phase, it's good to recall that several protocols have been designed in an attempt to secure the routing system but never implemented by major manufacturers as Nokia, Cisco or Juniper. For example, S-BGP, soBGP, psBGP. But as (IETF - Securing BGP and SIDR) states all of them contributed its' way to conform the currently most tracking standards that are getting standardized: RPKI and BGPSec.

Nowadays we may find around ~9% of valid prefixes in RPKI repositories (RPKI Monitor). This sets the base structure for BGPSec that has been formally released at late 2017. It's not possible to find publicly available implementations of this protocol inside standardized routing software suites yet. But (NIST) BGP-SRx offers a testing suite for this protocol.

On the **second phase** we focused on providing a comfortable way to deploy and change the configuration our testing environment. In addition, we focused and spent a lot of effort on installing and configuring tools that allow us to measure the impact of attacks over the whole network. More than concluding something, this phase demonstrated that lightweight complex environments can be deployed on home personal computers on the top of Linux KVM and we integrated a simple way using Elasticsearch, PacketBeat with some Python scripts plus Kibana to measure the performance and the impact of attacks on each case.

On **the third phase**, we developed practical examples of the attacks over constructed simulation environment and we improved the working environment security by going step by step through all protections outlined at (RFC7454 - BGP Operations and Security). Each step analyzed a protection, showed attacks performance against an unprotected environment, improved the automation developed on the previous phase and finally showed the security benefits provided by implementing the given protection. This phase is divided in two sub-phases.

The first of them analyzed the protections out of the scope of Secure Inter Domain Routing (SIDR) and showed the performance of several protections in front of different attacks. This phase demonstrated the benefits of hardening routers, using strong passwords to guarantee the origin of information exchanged between BGPv4 speakers and the benefits of implementing strict filter policies to manage what route advertisements are accepted or emitted, from and to each router. The problem is complicated. We also showed the performance of two of the most devastating attacks against a mesh of BGPv4 speakers, the prefix and sub-prefix hijacks.

We concluded that, the victim of both attacks has nothing to do against them since configuration mistakes on other points of the network makes the traffic be delivered to other hosts but to the legitimate party. In addition, we may think that combination of protections analyzed on this phase would theoretically protect against both this attacks on certain cases, but we could say that this case is pure theory, given the size of structures that should be maintained at all routers. The effort is not worth.

The second sub-phase analyzed the protections behind the scope of SIDR, that are currently supported by routing software suites, RPKI. This protocol provide prefix origin validation. On this phase we measure the benefits in front of prefix and sub-prefix hijacks, we configured a RPKI-RTR Cache server with set of Validated ROA Prefixes (VRPs) defining the routing policies for the testing environment and re-executed the prefix and sub-prefix hijacks. Both executions demonstrated that when configured in a secure manner, certificates are correctly issued, prefix and sub-prefix hijacks are not possible. We also executed a next hop hijack attack on certain configurations.

We concluded that when the full network is operating with RPKI and therefore, accepting only validated prefix advertisements, the prefix and sub-prefix hijacks are not possible. Partial and concretely low percentage of adoption deployments do not provide such benefit since parties that do not validate advertisements will keep exposing the rest of the network to this kind of attacks. The effect will spread as long as it reach a portion of the network validating against RPKI.

We concluded that, on high percentage of adoption, RPKI provides tangible security improvements.

But RPKI does not provide full protection to the Internet's routing system. Even on full deployments, we demonstrated that it possible for an attacker to execute a next hop hijack attack but the conditions to success are far more complicated than for executing prefix and sub-prefix hijacks. These conditions set is also supported by all protections explained on the previous sub-phase.

From this phase we concluded that each protection do its bit towards protecting the whole environment.

Fulfillment of objectives

At the beginning, the introduction section outlined to main objectives for the thesis:

1. To provide a practical and up-to-date analysis of security measures stated at (RFC7454 - BGP Operations and Security). For the subset that is supported by Quagga, Bird or Frrouting. Routing suites for Linux systems.
2. To deploy several virtualized scenarios to show the performance of protections in front of some attacks.

Starting from the latter objective, we generated an initial insecure simulation of the Internet and from this point we iterated defenses stated at (RFC7454 - BGP Operations and Security). We automated the deployment process, so it was easy for us to stack a protection on each iteration and to show the security benefit step by step.

(RFC7454 - BGP Operations and Security), states that implementing the following protections is recommended. The following table shows a summary of the work done for each protection stated:

Protection			**Work Done**	**Attacks / Analyses**
Speaker	Kernel		Adaptation of Linux TCP/IP Network stack behavior to operate as required.	Theoretical
	Firewall		Design and implementation a set of ACLs as required.	Theoretical
TCP Session	TCP-MD5		Configuration, deployment and performance analysis in front of the following attacks.	MITM
				Brute- force
	GTSM		We analyzed the BGPv4 daemon behavior in front of different IPv4 TTL values.	Behavior analysis
Routing	Prefix Filters		Design and deployment of a practical example. It showed the performance of attacks and the complexity of keeping up to date these filters.	Route Leak, Prefix and Sub-Prefix hijacks
	Route Flap Dampening		Flapping route generation. Theoretical analysis of the impact in both protected and unprotected environments.	Flapping Route
	Maximum Prefixes		Automatic generation of host direct routes (/32). Show the denial of service provoked and the need for the limit.	DoS – Resource Exhaustion
	AS Path filtering		Design and deployment of a practical example. This example showed the benefits of implementing this protection.	Route Leak, Prefix and Sub-Prefix hijacks
	Next hop filtering		Spoof of the next-hop and impact analysis. After that, we configured the routers to overwrite the next-hop advertised with advertiser address.	NextHop Hijack and Spoof
	SIDR	RPKI	Design and deployment of practical configuration examples. Evidence of the	Next-Hop Hijack

			benefits in front of prefix and sub-prefix hijacks.	
		BGPSEC	No actions have been taken to show the performance in front of Next-Hop Hijack.	N/A

From this summary, we can say that we achieved the objectives since we analyzed all protections supported nowadays by standardized routing software suites.

Further work

This section pretends to be self-critical and to show what points can be worked out in more depth.

Prefix Filters

We concluded that it's tough work to manage the volume and the complexity of the network nowadays to secure route exchanges through static configurations.

During the analysis, we did not make use of (IRRToolset) and neither studied (RFC2622 - Routing Policy Specification Language (RPSL)), (RFC4012 - Routing Policy Specification Language next generation (RPSLng)) or it's integration with RPKI, (RFC7909 - Securing Routing Policy Specification Language (RPSL) Objects).

IRRToolSet is capable of generating routing daemon configuration to adapt to business and technical needs of the network. It feeds from whois databases offering the publicly available routing policies.

We did not used it since we wanted to understand the concepts involved and not to take an already smart tool to do it for us. Anyway, in a future version of the simulation this could be of interest if the network grows up or if complex or real case peering agreements are to be simulated.

RPKI

We analyzed the integration and benefits of deploying and configuring RPKI. But we just did it on the routing side, we created a cache server and simulated the validated input ROAs attesting the validity of some VRPs, but we did not emphasize on the Resource Public Key Infrastructure, the trust anchors or certificate structure were merely mentioned and therefore, this opens a new line of research to improve the project. A good point to start this work would be the recently published (RFC8360 - Resource Public Key Infrastructure (RPKI) Validation Reconsidered)

BGPSEC

Since (RFC8205 - BGPsec Protocol Specification) requires routers to be upgraded with dedicated cryptographic processors it opens nice ways to extend the project. For example, to integrate (BGP Secure Routing Extensions). Afterwards, perform the practical analysis to show the performance against next-hop and the security benefits. In the meantime, it would be also interesting to analyze the protocol, it's integration with RPKI in order to have a better understanding of the most tracking alternative to secure the Internet routing system.

7. Bibliography

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

"Bitcoin Hijack: Routing Attacks on Cryptocurrencies." (2018). <https://btc-hijack.ethz.ch/>.

Cisco®. "Cisco RPKI." (n.d.). <https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-52/142-bgp.html>.

Debian. "Debian." (n.d.). <https://www.debian.org/>.

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

—. "Securing The BGPv4: PEC1 Schedule." 2018.

—. "Securing The BGPv4: PEC2 Security Analyses - Part 1." 2018.

—. "Securing The BGPv4: PEC3 Security Analyses - Part 2." 2018.

—. "Securing The BGPv4: Working Environment." 2018.

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

"IETF - Securing BGP and SIDR." (n.d.). <https://www.ietfjournal.org/securing-bgp-and-sidr/>.

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

ISC. "IRRToolset." (n.d.). <http://irrtoolset.isc.org>.

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

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

Lychev, Robert, Schapira Michael y Sharon Goldberg. «Rethinking Security for Internet Routing.» 2016.

MANRS. "Another BGP Hijacking Event Highlights the Importance of MANRS and Routing Security." 2018. <https://www.manrs.org/2018/04/another-bgp-hijacking-event-highlights-the-importance-of-manrs-and-routing-security/>.

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

NIST. "BGP Secure Routing Extensions." (n.d.). <https://www.nist.gov/services-resources/software/bgp-secure-routing-extension-bgp-srx-prototype>.

—. "RPKI Monitor." (n.d.). <https://rpki-monitor.antd.nist.gov/>.

"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.

"RFC2622 - Routing Policy Specification Language (RPSL)." (n.d.). <https://tools.ietf.org/html/rfc2622>.

"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.

"RFC6487 - A Profile for X.509 PKIX Resource Certificates." n.d. <https://tools.ietf.org/html/rfc6487>.

"RFC6488 - Signed Object Template for the Resource Public Key Infrastructure (RPKI)." 2012.

"RFC6493 - The Resource Public Key Infrastructure (RPKI) Ghostbusters Record." 2012. <https://tools.ietf.org/html/rfc6493>.

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

"RFC6811 - BGP Prefix Origin Validation." 2013.

"RFC6841 - A Profile for Resource Certificate Repository Structure." 2012. <https://tools.ietf.org/html/rfc6481>.

"RFC6842 - A Profile for Route Origin Authorizations (ROAs)." n.d. <https://tools.ietf.org/html/rfc6482>.

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

"RFC7454 - BGP Operations and Security." 2015.

"RFC7454 - BGP Operations and Security." 2015.

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

"RFC7909 - Securing Routing Policy Specification Language (RPSL) Objects." (2016). <https://tools.ietf.org/html/rfc7909>.

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

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

"RFC8205 - BGPsec Protocol Specification." 2017.

"RFC8360 - Resource Public Key Infrastructure (RPKI) Validation Reconsidered." (2018). <https://tools.ietf.org/html/rfc8360>.

"RPKI-RTR-Server." n.d. <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>.