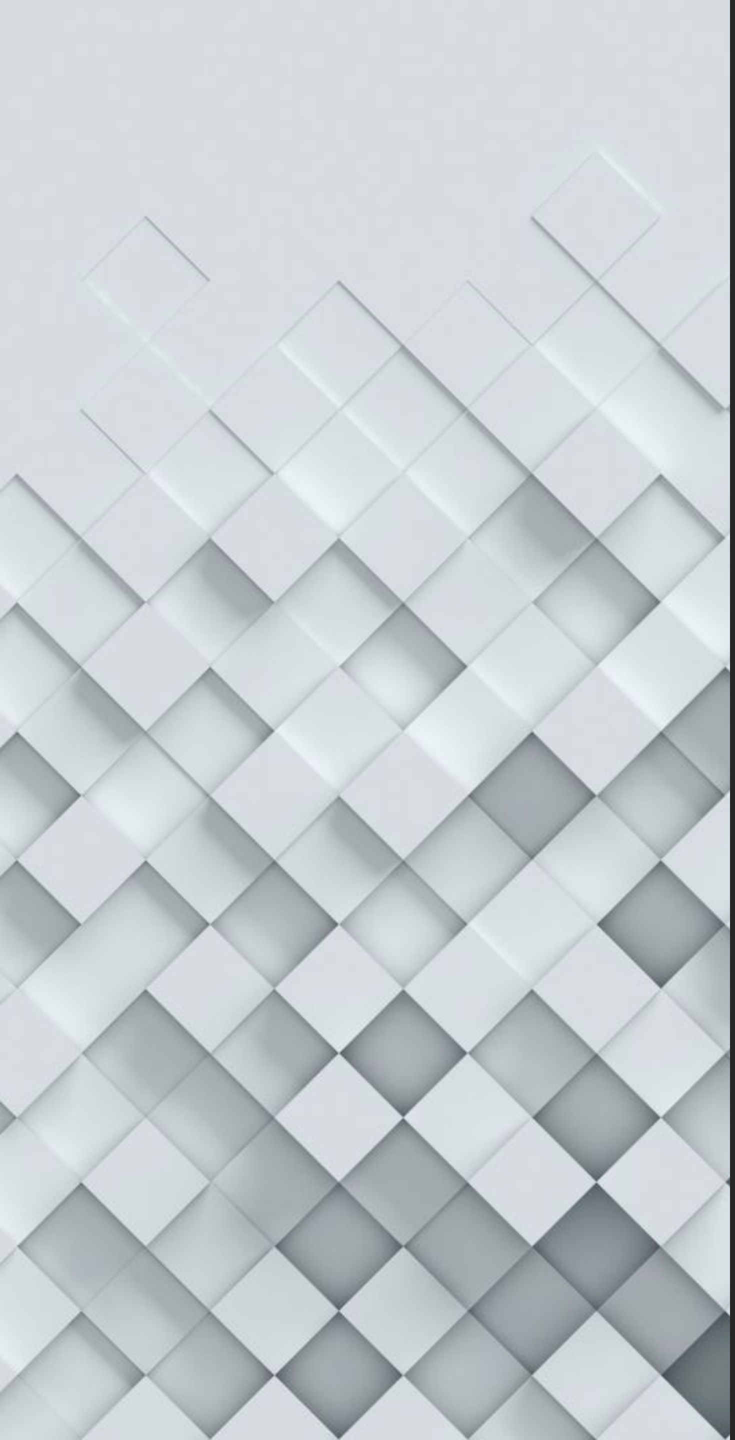


BGP-ASPA Hackathon Report

IETF 112

Hackathon Team:
O. Borchert, L. Hannachi, D. Montgomery, K. Sriram

Goal

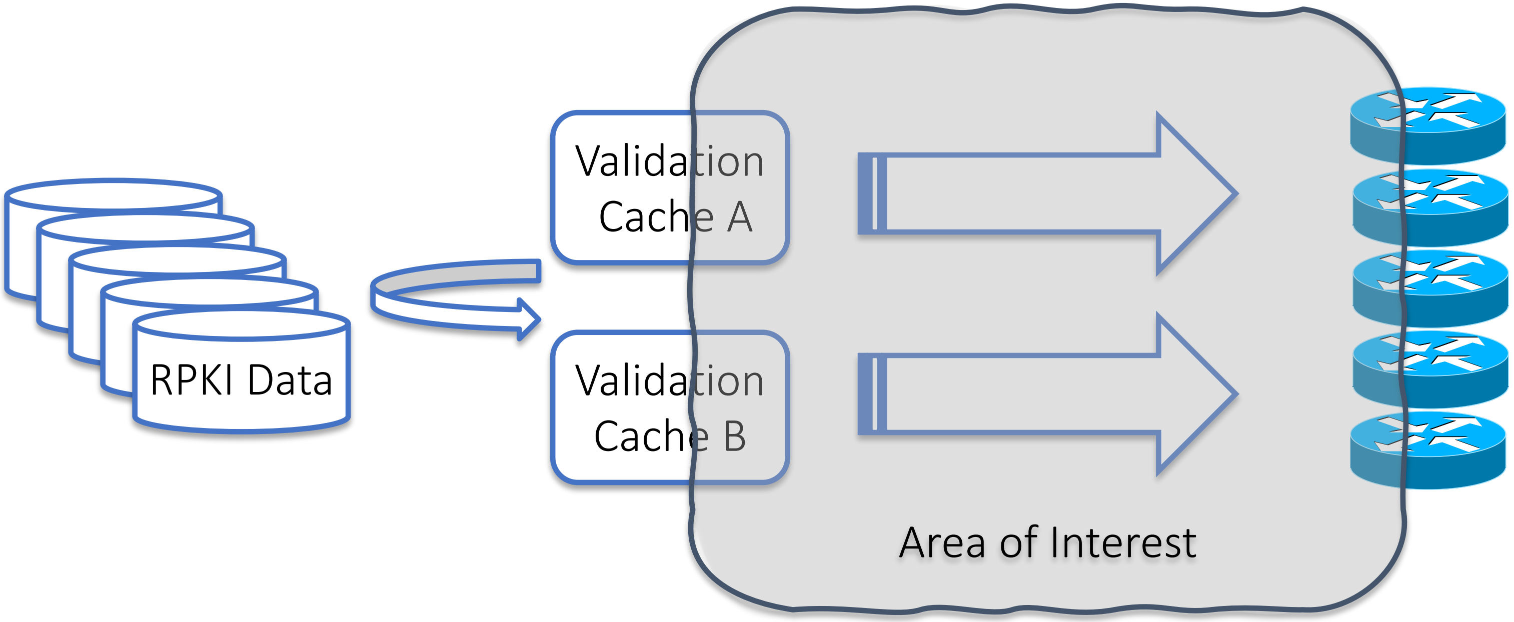
- Develop tools and data sets to facilitate testing emerging BGP route-leak mitigation technique

What tools did we use?

- We use the NIST BGP-SRx Software Suite V6 that provides reference implementation for:
  - draft-ietf-sidrops-aspa-verification-08+ (update with algorithm correction*)
  - draft-ietf-sidrops-8210bis-03
  - and test harnesses that enables scripted experiments with RPKI & BGP data sets.
- Source: https://github.com/usnistgov/NIST-BGP-SRx

* https://datatracker.ietf.org/meeting/110/materials/slides-110-sidrops-sriram-aspa-alg-accuracy-01

High Level ASPA Data Flow



RPKI Data

Validation Cache A

Validation Cache B

Area of Interest

Tasks

Develop tools and data sets for testing router implementations of ASPA. Unit tests and Internet scale tests

Task 1:

- Create sample Internet scale ASPA data set for use with 8210bis-03 using CAIDA reference data
- Use SRx Test harness ASCII format:

  addASPA <AFI> <CustomerAS> <ProviderAS>+

Task 2:

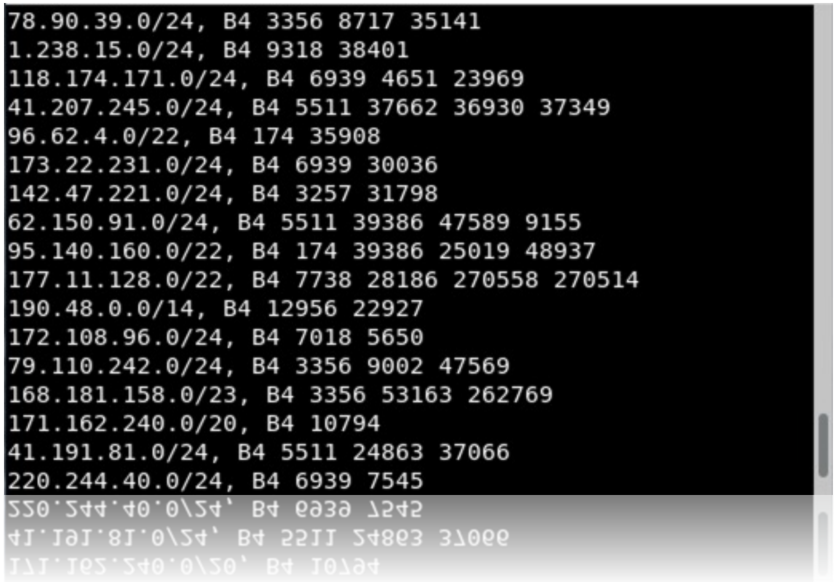
- Create sample BGP UPDATES using data from RouteViews3.

What got done

- We designed a test framework that allows to generate
  - CAIDA based ASPA script data describing 72, 616 ASPA PDU’s containing 148,284 customer provider relations.
  - Data Pool is down-selectable to only use ASPA link relations for ASN’s found within UPDATE stream only
  - Specified a result output that can be used to compare between implementations
- Created Data Sets 100, 500, 800, 1K, 10K, and 20K unique AS PATHs using RouteViews and CAIDA Data

1. Preparation of BGP peers from RouteViews3 Data Set for BGPsec-IO

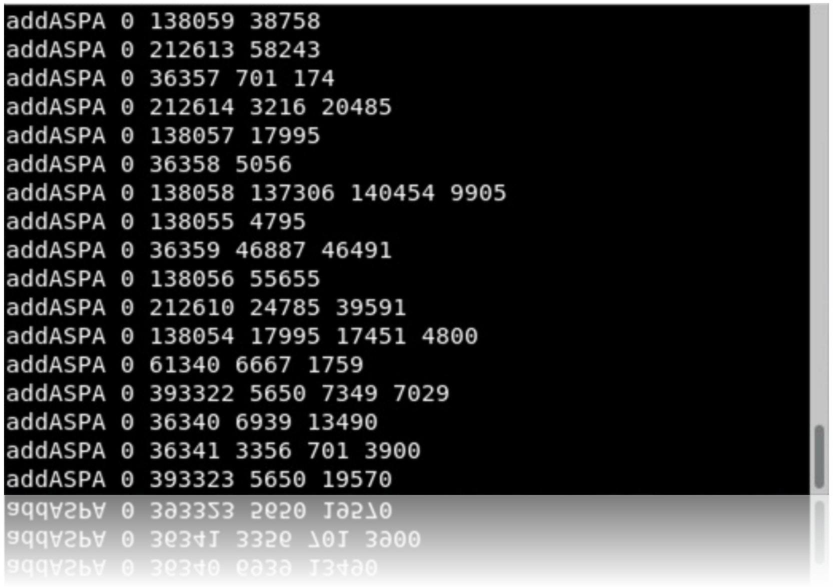
- We generated UPDATE traffic files, one for each peer containing the UPDATE send to the collector
- We removed the Peer AS (will be added by the player again)
- We added the marker B4 BGPsec-IO to only generate BGP-4 UPDATES and NOT BGPsec UPDATES



```
78.90.39.0/24, B4 3356 8717 35141
1.238.15.0/24, B4 9318 38401
118.174.171.0/24, B4 6939 4651 23969
41.207.245.0/24, B4 5511 37662 36930 37349
96.62.4.0/22, B4 174 35908
173.22.231.0/24, B4 6939 30036
142.47.221.0/24, B4 3257 31798
62.150.91.0/24, B4 5511 39386 47589 9155
95.140.160.0/22, B4 174 39386 25019 48937
177.11.128.0/22, B4 7738 28186 270558 270514
190.48.0.0/14, B4 12956 22927
172.108.96.0/24, B4 7018 5650
79.110.242.0/24, B4 3356 9002 47569
168.181.158.0/23, B4 3356 53163 262769
171.162.240.0/20, B4 10794
41.191.81.0/24, B4 5511 24863 37066
220.244.40.0/24, B4 6939 7545
```

Convert all CAIDA Data to BGP-SRx Cache Test Harness Format

- To use the “rpkirtr_svr” BGP-SRx cache test harness we needed the CAISA data in the following script style: addASPA <afi> <customer> <provider>+
- We generated a total of 72,616 ASPA data entries with 148,284 link relations.

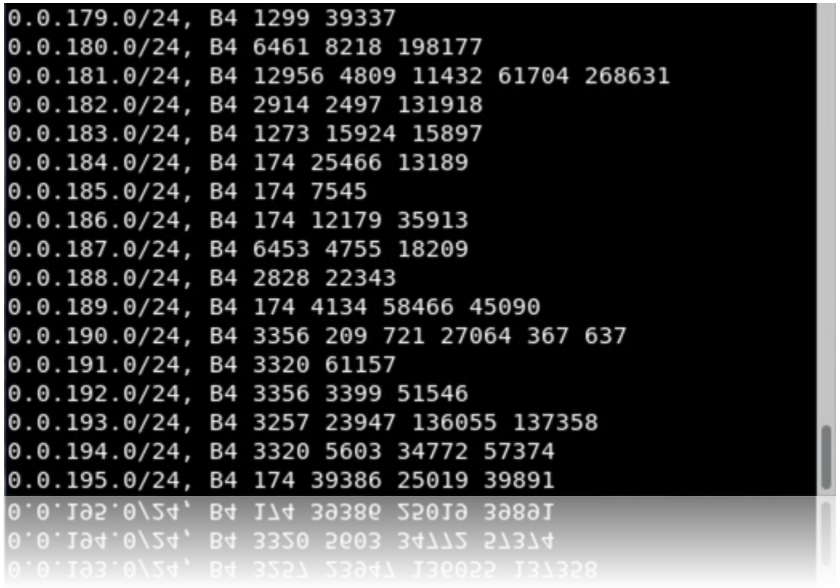


```
addASPA 0 138059 38758
addASPA 0 212613 58243
addASPA 0 36357 701 174
addASPA 0 212614 3216 20485
addASPA 0 138057 17995
addASPA 0 36358 5056
addASPA 0 138058 137306 140454 9905
addASPA 0 138055 4795
addASPA 0 36359 46887 46491
addASPA 0 138056 55655
addASPA 0 212610 24785 39591
addASPA 0 138054 17995 17451 4800
addASPA 0 61340 6667 1759
addASPA 0 393322 5650 7349 7029
addASPA 0 36340 6939 13490
addASPA 0 36341 3356 701 3900
addASPA 0 393323 5650 19570
```

Creation of Test Traffic

We specify the peer and the maximum UPDATES

- Here we down select the peers UPDATES to “X” UNIQUE AS Paths and removed the prefix.
- We added a synthetic generated prefix from the prefix pool 0.0.1.0/24 to 255.255.255.0/24 to assure no path uses the same prefix.*



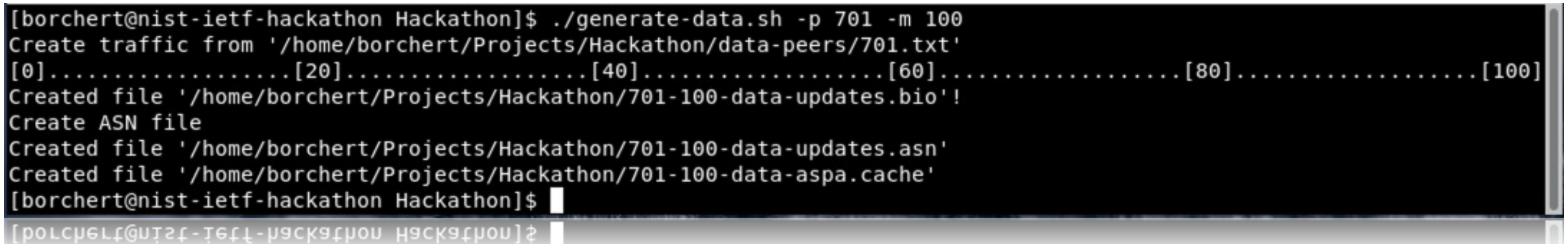
```
0.0.179.0/24, B4 1299 39337
0.0.180.0/24, B4 6461 8218 198177
0.0.181.0/24, B4 12956 4809 11432 61704 268631
0.0.182.0/24, B4 2914 2497 131918
0.0.183.0/24, B4 1273 15924 15897
0.0.184.0/24, B4 174 25466 13189
0.0.185.0/24, B4 174 7545
0.0.186.0/24, B4 174 12179 35913
0.0.187.0/24, B4 6453 4755 18209
0.0.188.0/24, B4 2828 22343
0.0.189.0/24, B4 174 4134 58466 45090
0.0.190.0/24, B4 3356 209 721 27064 367 637
0.0.191.0/24, B4 3320 61157
0.0.192.0/24, B4 3356 3399 51546
0.0.193.0/24, B4 3257 23947 136055 137358
0.0.194.0/24, B4 3320 5603 34772 57374
0.0.195.0/24, B4 174 39386 25019 39891
```

* Can happen if raw data comes from UPDATE stream and not RIB in.

Creation of APSA Test Data

The ASPA data is generated depending on the UPDATE traffic.

- From the selected UPDATE traffic a list of all unique ASes is generated
- From the 72K available Customer specification only those ASes are selected that found in the UPDATE traffic.
- A downsized ASPA data file is generated

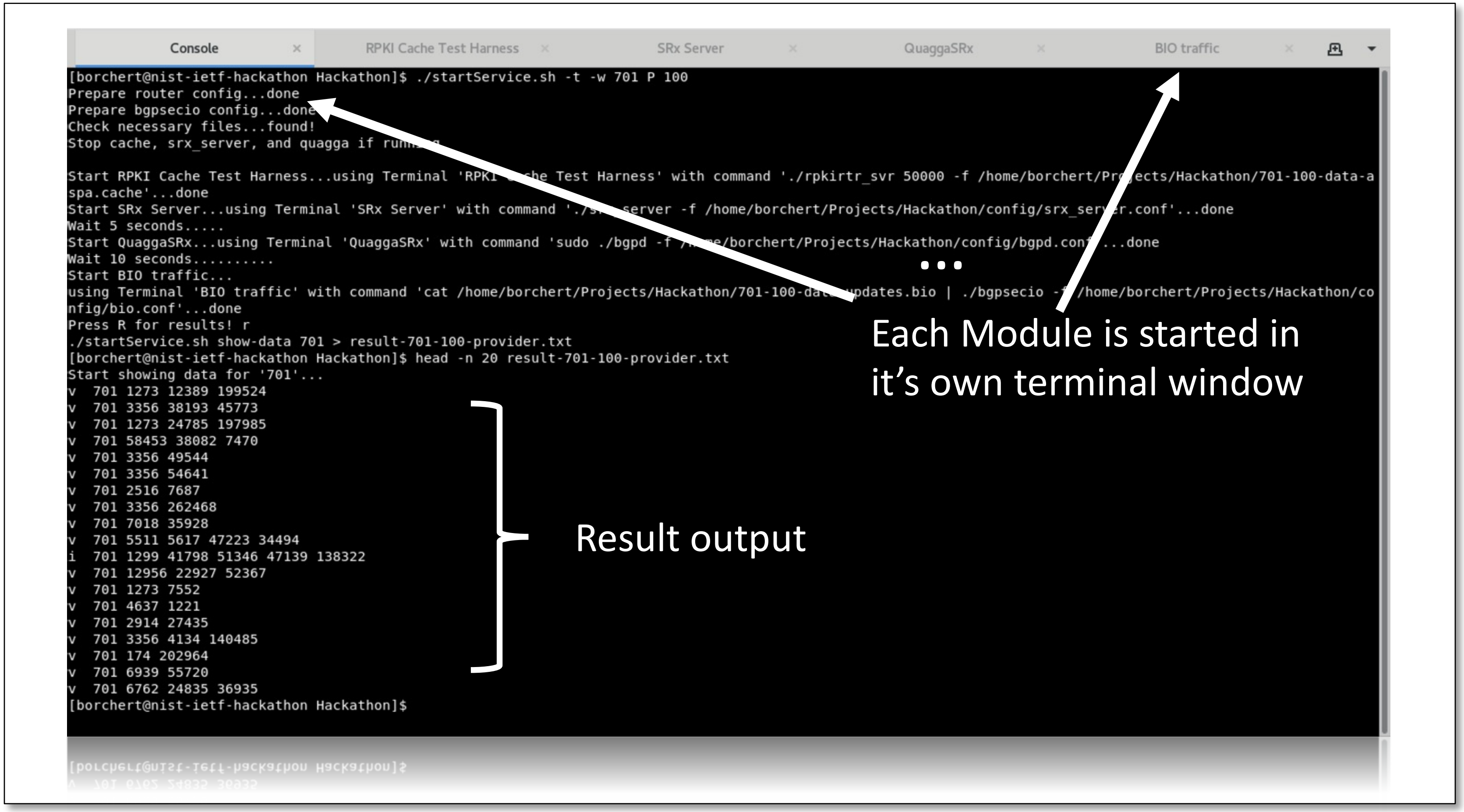


```
[borchert@nist-ietf-hackathon Hackathon]$ ./generate-data.sh -p 701 -m 100
Create traffic from '/home/borchert/Projects/Hackathon/data-peers/701.txt'
[0]...................[20]...................[40]...................[60]...................[80]...................[100]
Created file '/home/borchert/Projects/Hackathon/701-100-data-updates.bio'!
Create ASN file
Created file '/home/borchert/Projects/Hackathon/701-100-data-updates.asn'
Created file '/home/borchert/Projects/Hackathon/701-100-data-aspa.cache'
[borchert@nist-ietf-hackathon Hackathon]$ █
```

Starting the Experiment

- Once the experimental data is generated, the starter script allows two modes:
  - Terminal Only
    - In this mode each module is started in the background
    - All output standard and error is redirected into log files.
  - Gnome Terminal
    - This mode is preferable for window based Linux systems
    - Here each module will be started in its own terminal tab
      - In case something goes wrong, this mode is simpler to debug.
      - This mode allows to control the cache test harness

The Gnome Terminal Mode



```
[borchert@nist-ietf-hackathon Hackathon]$ ./startService.sh -t -w 701 P 100
Prepare router config...done
Prepare bgpsecio config...done
Check necessary files...found!
Stop cache, srx_server, and quagga if running

Start RPKI Cache Test Harness...using Terminal 'RPKI Cache Test Harness' with command './rpkirtr_svr 50000 -f /home/borchert/Projects/Hackathon/701-100-data-a
spa.cache'...done
Start SRx Server...using Terminal 'SRx Server' with command './srx_server -f /home/borchert/Projects/Hackathon/config/srx_server.conf'...done
Wait 5 seconds.....
Start QuaggaSRx...using Terminal 'QuaggaSRx' with command 'sudo ./bgpd -f /home/borchert/Projects/Hackathon/config/bgpd.conf'...done
Wait 10 seconds..........
Start BIO traffic...
using Terminal 'BIO traffic' with command 'cat /home/borchert/Projects/Hackathon/701-100-data-updates.bio | ./bgpsecio -f /home/borchert/Projects/Hackathon/co
nfig/bio.conf'...done
Press R for results! r
./startService.sh show-data 701 > result-701-100-provider.txt
[borchert@nist-ietf-hackathon Hackathon]$ head -n 20 result-701-100-provider.txt
Start showing data for '701'...
v  701 1273 12389 199524
v  701 3356 38193 45773
v  701 1273 24785 197985
v  701 58453 38082 7470
v  701 3356 49544
v  701 3356 54641
v  701 2516 7687
v  701 3356 262468
v  701 7018 35928
v  701 5511 5617 47223 34494
i  701 1299 41798 51346 47139 138322
v  701 12956 22927 52367
v  701 1273 7552
v  701 4637 1221
v  701 2914 27435
v  701 3356 4134 140485
v  701 174 202964
v  701 6939 55720
v  701 6762 24835 36935
[borchert@nist-ietf-hackathon Hackathon]$
```

…

Each Module is started in it’s own terminal window

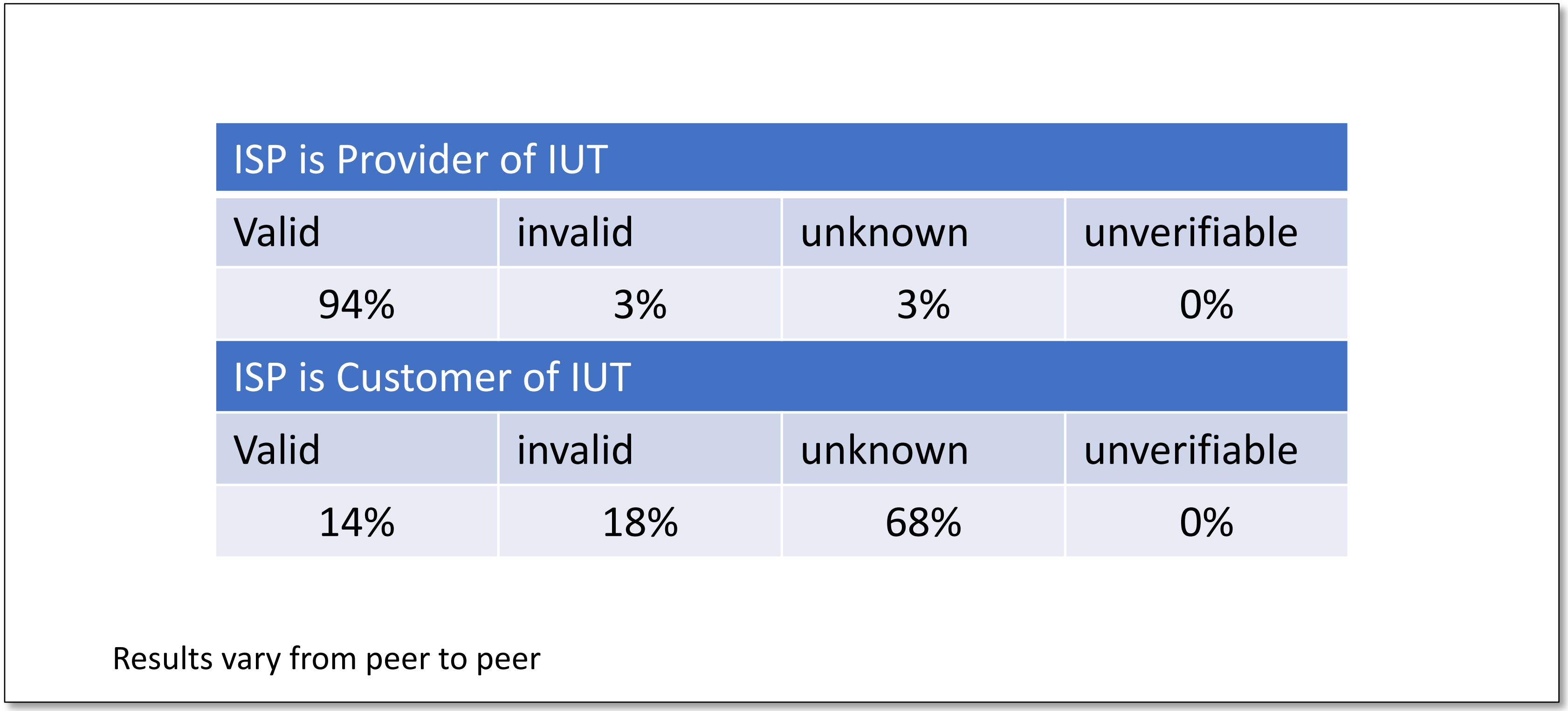
Result output

The Experiment

We used RouteViews-3 BGP data, Large Scale ISP and CAIDA data from Oct. 1, 2020

- We created a subset of unique routes.
- We selected only CAIDA data where ASN in each path is listed as customer
- Then we performed ASPA validation
- IUT is private ASN peering with Large Scale ISP

Some Results



ISP is Provider of IUT			
Valid	invalid	unknown	unverifiable
94%	3%	3%	0%
ISP is Customer of IUT			
Valid	invalid	unknown	unverifiable
14%	18%	68%	0%

Results vary from peer to peer

The Code

- We still refine the code and then will publish it once its ready
  - Once published we will provide the location of the framework on the list
  - Also we will provide a link in out GitHub page for NIST BGP-SRx V6:

Future Work / Hackathons

- More experiments to study gradual deployment of ASPA objects
    - Selecting different peers
- For proper performance testing extending framework to use multiple peering sessions
    - Manual possible but it would be nice to have it automated as well
    - Scaling, scaling, scaling,….
- Other implementations to test against
    - Maybe next hackathon
- Create ASPA objects for testing Validation Caches?
    - Maybe others can join in!

Questions ?

Oliver Borchert
oliver.borchert@nist.gov

General Questions:
itrg-contact@list.nist.gov