

INTENTIONET

YOUR NETWORK. EXACTLY AS YOU INTENDED.

Batfish: Make BGP route-leaks history

February 2020

Samir Parikh

samir@intentionet.com

www.intentionet.com

What is Batfish?



Pre-deployment network validation solution

Started as a Microsoft Research project in 2012

- Open sourced in 2014 (Apache 2.0)
- Contributors from Intentionet, BBN, Colgate University, Microsoft, Princeton, UCLA, USC, ...

Used by many Fortune 500 companies

A screenshot of the GitHub repository page for batfish/batfish. The page shows the repository name, navigation tabs for Code, Issues (116), Pull requests (9), Wiki, and Insights. A description of Batfish as a network configuration analysis tool is visible, along with tags for network, configuration, configuration-parser, configuration-analysis, and network-verification. At the bottom, it shows 6,295 commits, 87 branches, and 52 forks.

batfish / batfish

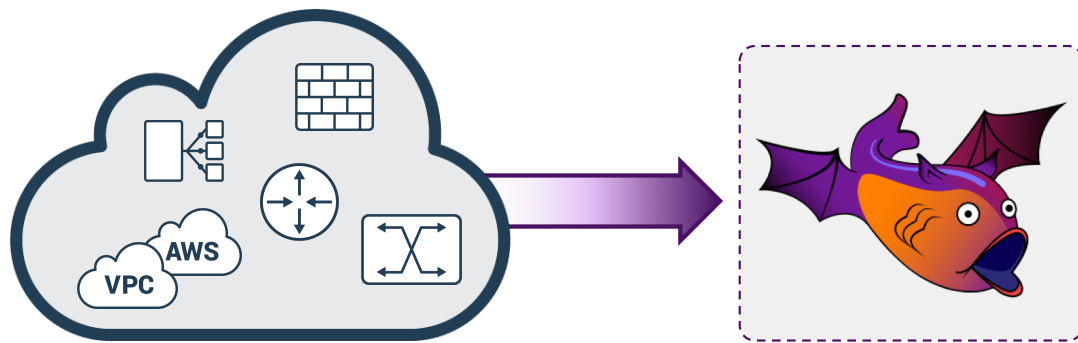
<> Code Issues 116 Pull requests 9 Wiki Insights

Batfish is a network configuration analysis tool that can find bugs and guarantee the network configurations. It enables network engineers to rapidly and safely evolve the security breaches. <http://www.batfish.org>

network configuration configuration-parser configuration-analysis network-verification

6,295 commits 87 branches 52 r

How It Works



INTENTIONET

How It Works

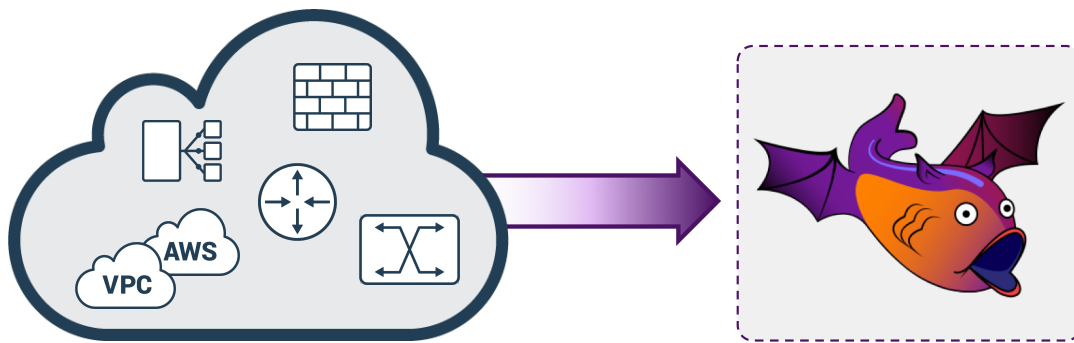
CUMULUS



*And many
more...*

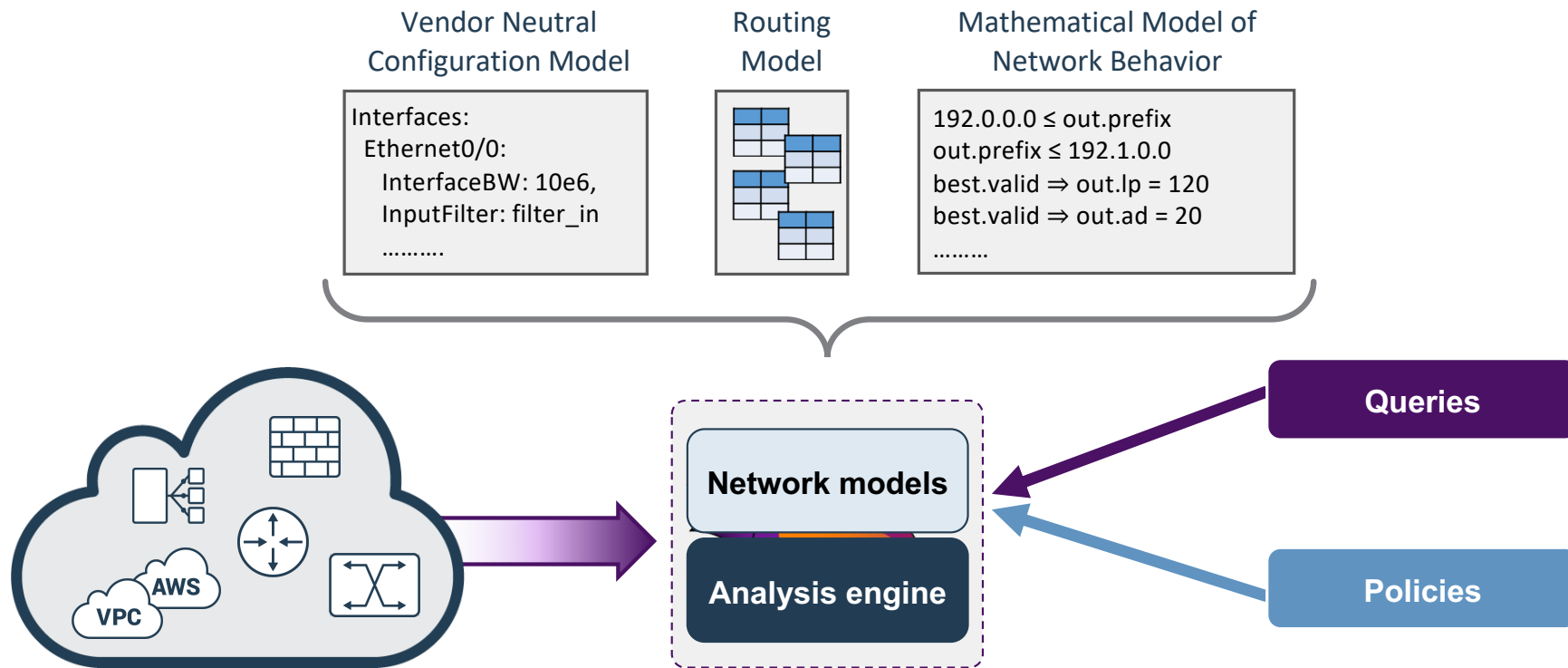
JUNIPER
NETWORKS

ARISTA



INTENTIONET

How It Works



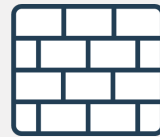
Batfish Policies and Queries

Configuration Audit



- Are all devices compliant with site standards?
- Are all protocol sessions (BGP, IPsec, MLAG, ...) compatible?

Comprehensive Firewall/ACL Analysis



- Is my firewall protecting sensitive services?
- What is the impact of this ACL change?

Route and Forwarding Analysis



- Is my network redundant?
- What happens if I change this route policy?

Comprehensive Reachability Analysis

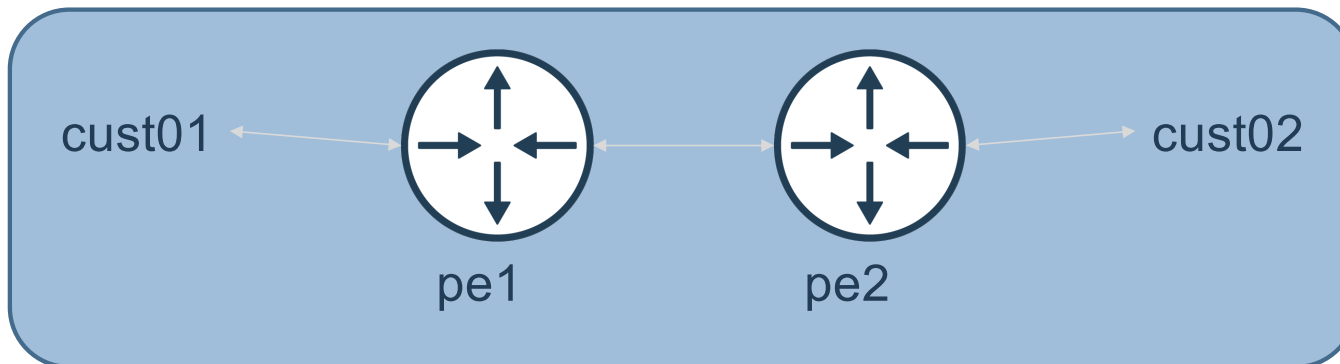


- Can any flow violate cross-site isolation?
- Is the DNS server accessible from anywhere?

Live Demonstrations

- 1) Leverage Batfish for continuous configuration validation
- 2) Analyze a change to a routing policy
- 3) Identify and prevent potential route-leak

DEMO #1 – Continuous configuration validation



DEMO #1 – Continuous configuration validation

- This demo uses Batfish to build an test suite for multi-vendor configuration validation
- Tests inspired by MANRS guidelines
<https://github.com/manrs-tools/MANRS-validator>



Mutually Agreed Norms for Routing Security (MANRS) is a global initiative, supported by the Internet Society, that provides crucial fixes to reduce the most common routing threats.

DEMO #1 – Continuous configuration validation

Vendor agnostic policies

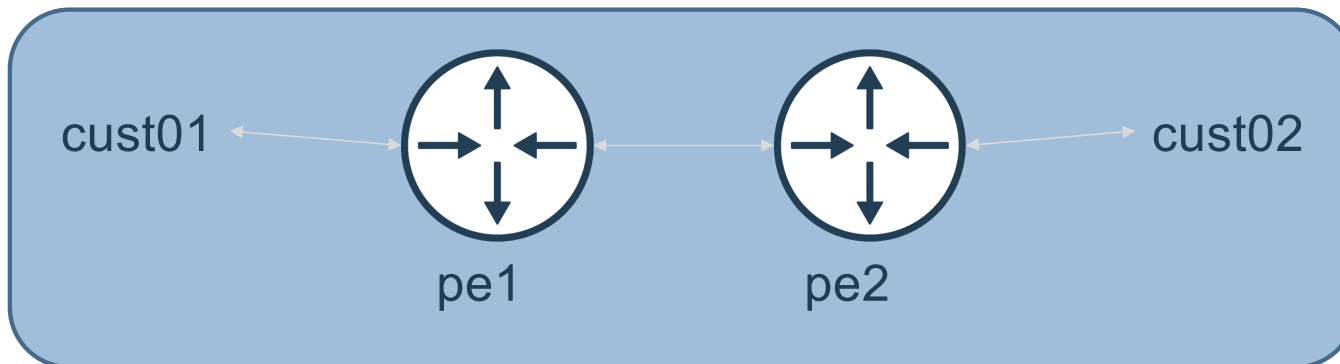
Example

Ensure all customer BGP sessions have an input policy

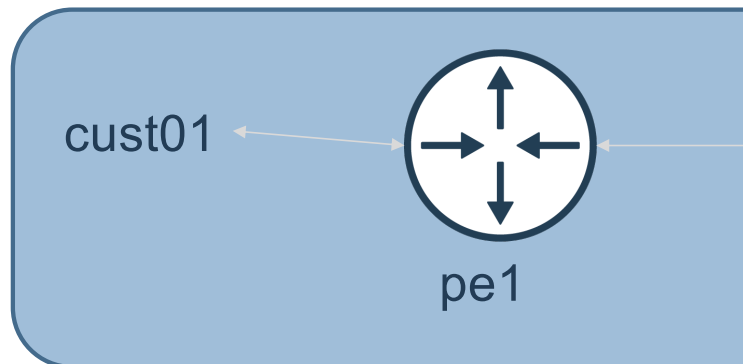
INTENTIONET

```
def test_customer_bgp_session_input_policy(bf, customer_list):  
    """Ensure all customer BGP peering sessions have INPUT policy configured"""  
    bf.asserts.current_assertion = 'Assert all customer BGP sessions have input route policy'  
  
    nodes = [] # determine the list of peering nodes which need to be evaluated  
    for customer in customer_list:  
        nodes.append(customer['Node'])  
    nodespec = ','.join(nodes)  
  
    # retrieve the BGP session configuration for all peers on the peering nodes  
    df = bf.q.bgpPeerConfiguration(nodes=nodespec).answer().frame()  
    bad_peers_in = []  
  
    for customer in customer_list:  
        # check the BGP session configuration for specific peers and  
        # extract input routing policy  
        iPol = df[(df['Node'] == customer['Node']) & \  
                (df['Remote_IP'] == customer['Remote_IP'])]['Import_Policy']  
        if len(iPol.iloc[0]) == 0:  
            bad_peers_in.append(f'{customer["Node"]}:{customer["Remote_IP"]}')  
  
    test = (len(bad_peers_in) == 0)
```

DEMO #2 – Test a routing policy change



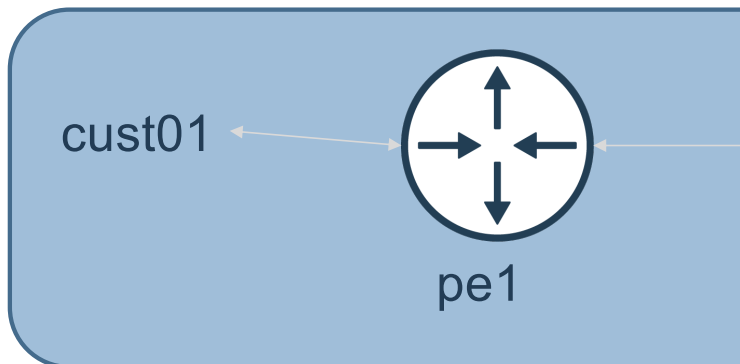
DEMO #2 – Test a routing policy change



PE Input policy for cust01

```
ip as-path access-list customer1 permit ^609
!  
ip prefix-list customer1 deny 10.0.0.0/8 le 32  
ip prefix-list customer1 deny 0.0.0.0/0  
ip prefix-list customer1 deny 127.0.0.0/8 le 32  
ip prefix-list customer1 deny 0.0.0.0/8 le 32  
ip prefix-list customer1 permit 0.0.0.0/0 le 32  
!  
ip community-list expanded customer1 permit _609:.*_  
!  
route-map customer1-in permit 10  
  match as-path customer1  
  match ip address prefix-list customer1  
  match community customer1  
  set community 609:1  
!  
route-map customer1-in deny 20
```

DEMO #2 – Test a routing policy change



Update policy to block /24 prefixes

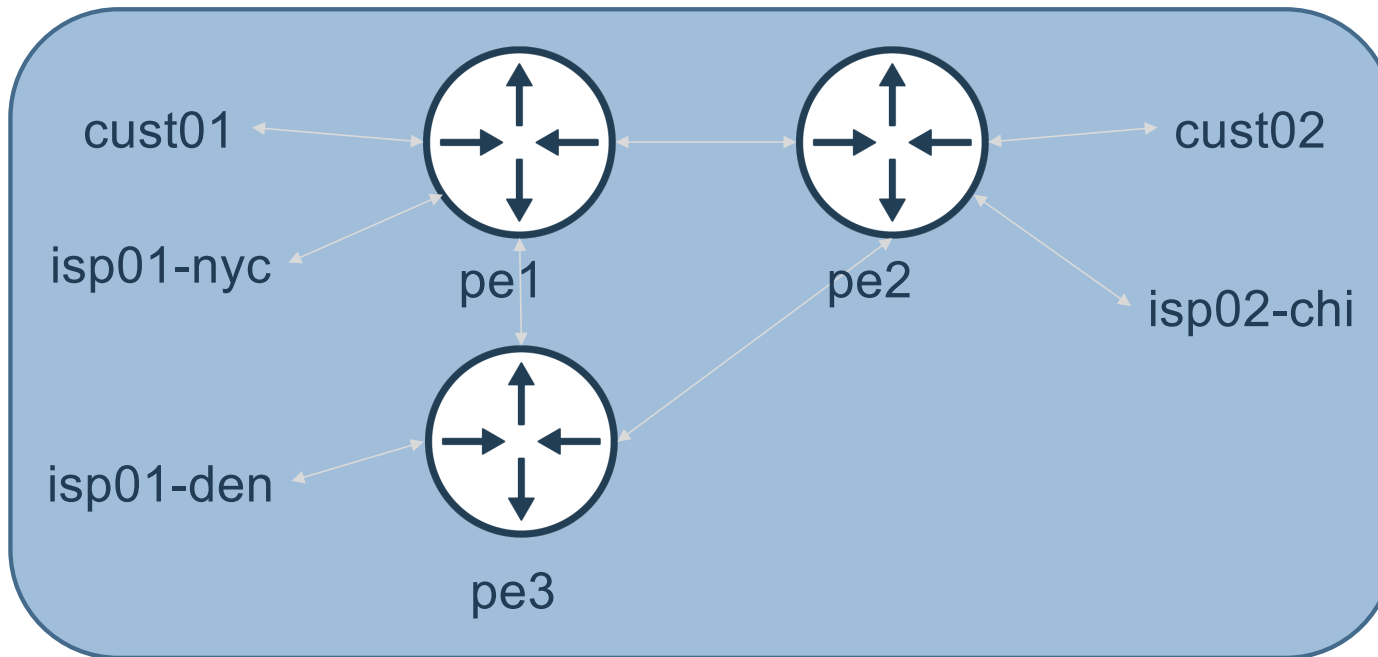
```
ip as-path access-list customer1 permit ^609
!  
ip prefix-list customer1 deny 10.0.0.0/8 le 32  
ip prefix-list customer1 deny 0.0.0.0/0  
ip prefix-list customer1 deny 127.0.0.0/8 le 32  
ip prefix-list customer1 deny 0.0.0.0/8 le 32  
ip prefix-list customer1 permit 0.0.0.0/0 le 32  
!  
ip prefix-list BLOCK24 permit 0.0.0.0/0 ge 24  
!  
ip community-list expanded customer1 permit _609:.*_  
!  
route-map customer1-in deny 10  
match ip address prefix-list BLOCK24  
!  
route-map customer1-in permit 20  
match as-path customer1  
match ip address prefix-list customer1  
match community customer1  
set community 609:1 additive  
!  
route-map customer1-in deny 30
```

DEMO #2 – Test a routing policy change

Process to analyze routing policy change

- 1) Collect **BGP-Adj-RIB-In** for peer in question
- 2) Upload snapshot with current and proposed policy to Batfish
- 3) Compare Batfish routing policy evaluation between current and proposed policy

DEMO #3 – Prevent a route-leak



DEMO #3 – Prevent a route-leak

Scenario

- Peer AS isp01 wants to send prefixes of length $>/24$ to load-balance traffic across peering links

Validation Steps

1. Collect **BGP-Adj-RIB-In** and verify that existing policy blocks $>/24$ prefixes
2. Update policy and test to verify that policy accepts prefixes
3. Validate that prefixes are not exported out of your AS
4. Repeat 3 & 4 until policy is correct

Getting Started with Batfish is easy!

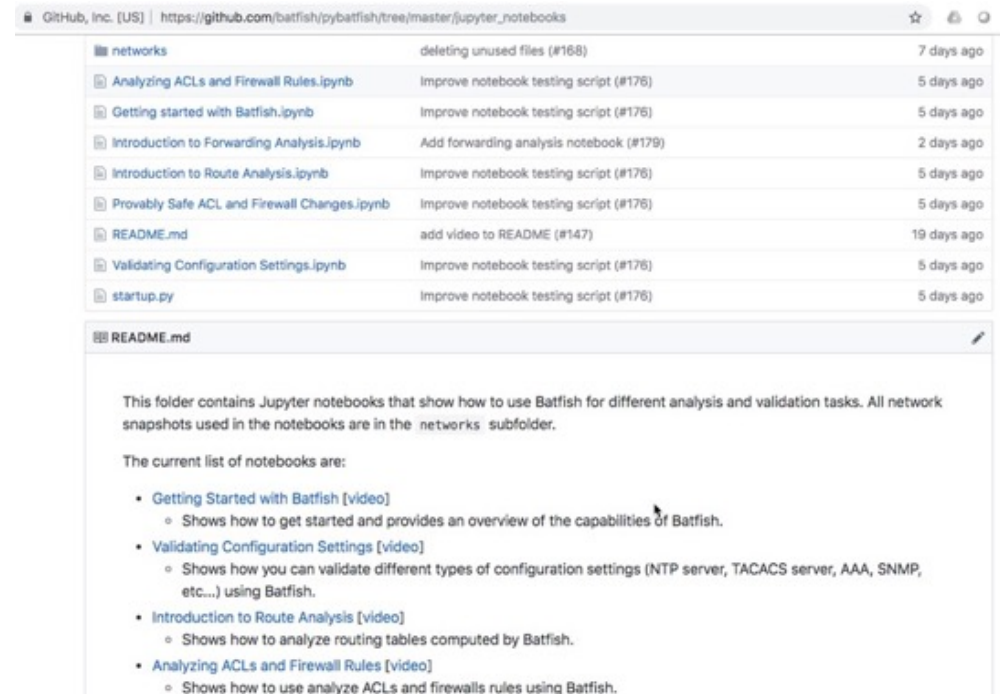
One line with Docker

https://batfish.readthedocs.io/en/latest/getting_started.html#installing-batfish

Advanced tutorials & videos

[https://www.github.com/batfish/pybatfish/
tree/master/jupyter_notebooks](https://www.github.com/batfish/pybatfish/tree/master/jupyter_notebooks)

[https://www.youtube.com/channel/
UCA-OUW_3lOt9U_s60KvmJYA](https://www.youtube.com/channel/UCA-OUW_3lOt9U_s60KvmJYA)



The screenshot shows a GitHub repository page for batfish/pybatfish. The top part displays a list of pull requests with columns for the pull request title, the description of the changes, and the time since the pull request was created. Below the pull requests, the README.md file is open, showing the following content:

This folder contains Jupyter notebooks that show how to use Batfish for different analysis and validation tasks. All network snapshots used in the notebooks are in the `networks` subfolder.

The current list of notebooks are:

- Getting Started with Batfish [video]
 - Shows how to get started and provides an overview of the capabilities of Batfish.
- Validating Configuration Settings [video]
 - Shows how you can validate different types of configuration settings (NTP server, TACACS server, AAA, SNMP, etc...) using Batfish.
- Introduction to Route Analysis [video]
 - Shows how to analyze routing tables computed by Batfish.
- Analyzing ACLs and Firewall Rules [video]
 - Shows how to use analyze ACLs and firewalls rules using Batfish.

Download the presentation and demo

<https://github.com/saparikh/nanog78-demo>

Validate your BGP policies with Batfish!



batfish-org



@batfish



batfish.org



A server rack with a blue glow and digital data overlays. The text "Thank You" is centered on the left side. The background shows server racks with various indicators and data points, including the number "1952" and the text "LAN0 DL0E".

Thank You

INTENTIONET
YOUR NETWORK. EXACTLY AS YOU INTENDED.