
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?

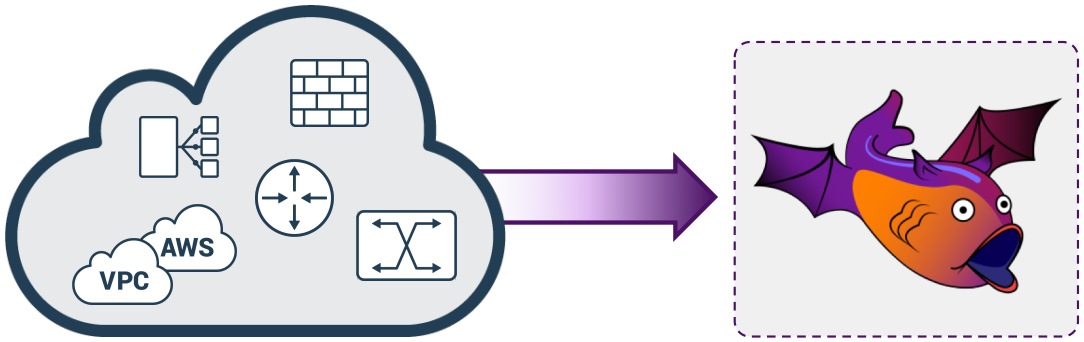
Pre-deployment network validation solution

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, …

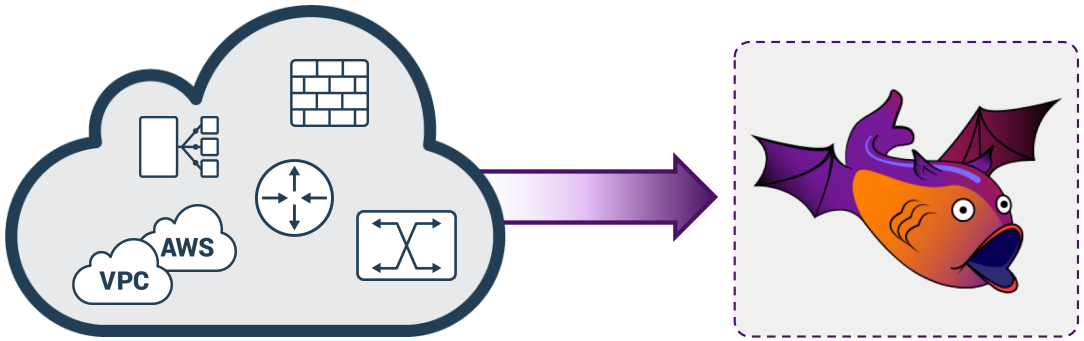
Used by many Fortune 500 companies

How It Works



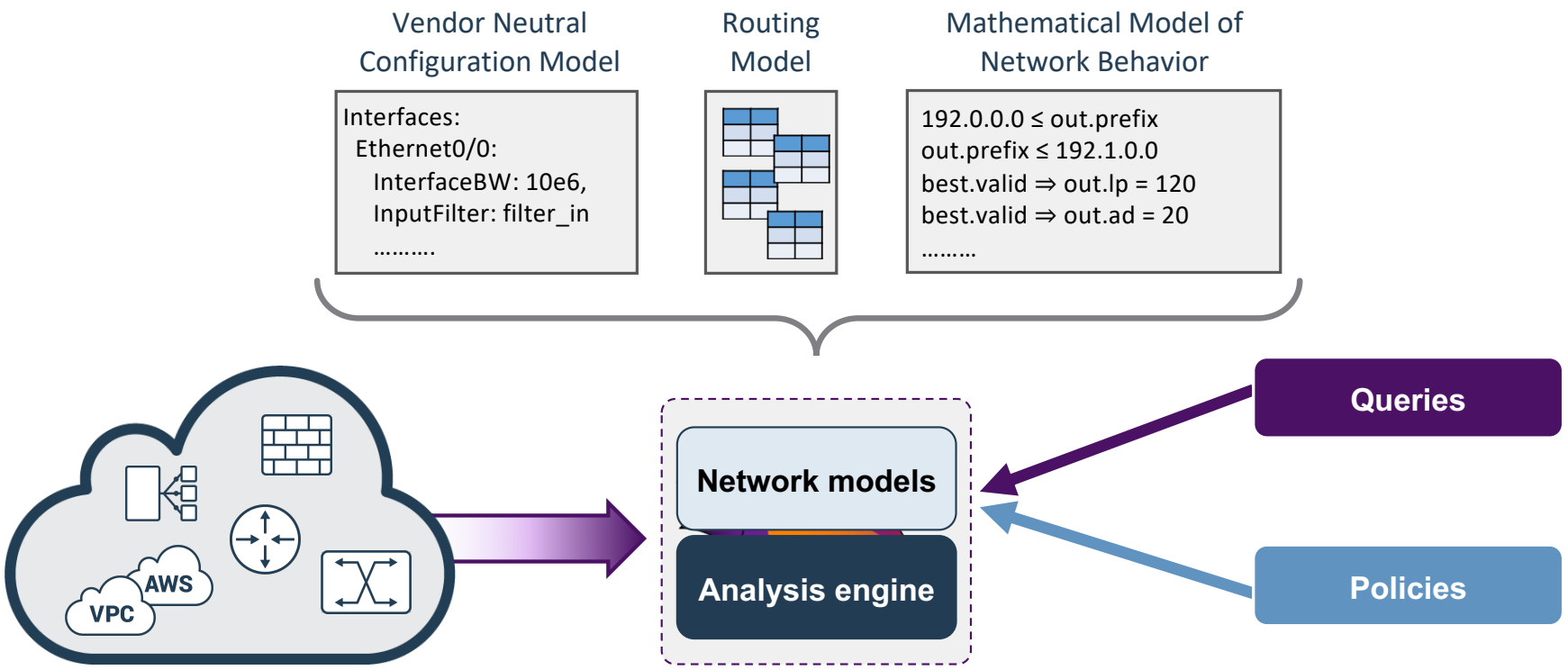
INTENTIONET

How It Works



INTENTIONET

How It Works



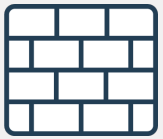
INTENTIONET

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

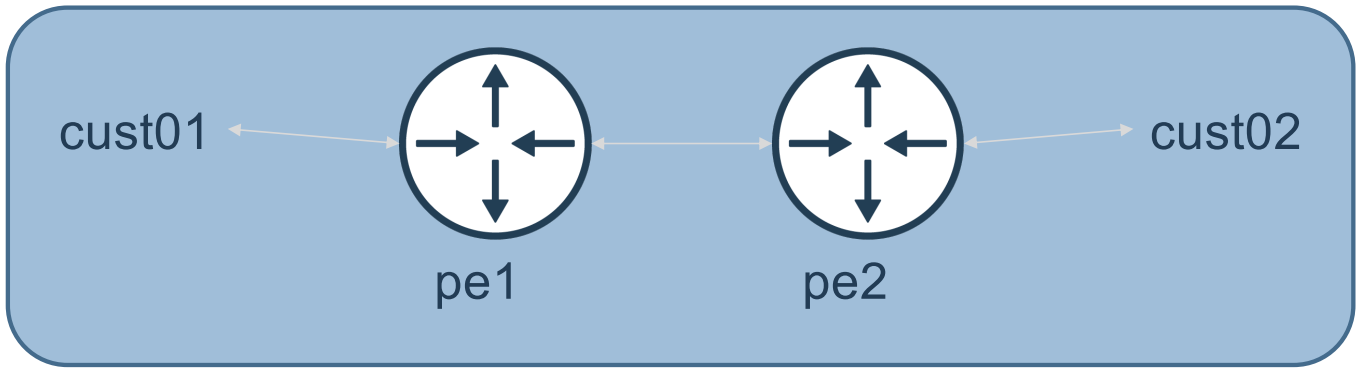
- Can any flow violate cross-site isolation?
- Is the DNS server accessible from anywhere?

INTENTIONET

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

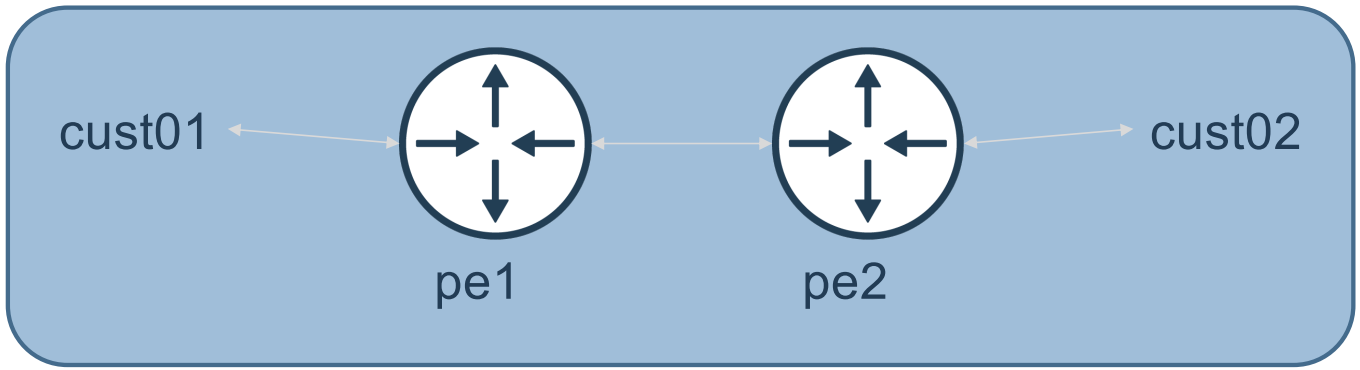
Example

Ensure all customer BGP sessions have an input policy

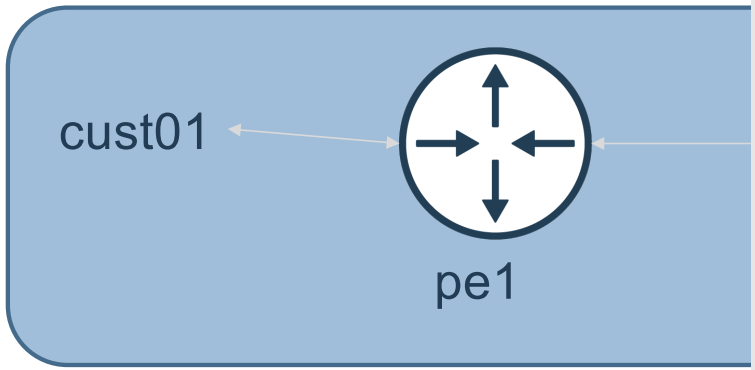
```
def test_customer_bgp_session_input_policy(bf, customer_list):
    """Ensure all customer BGP peering sessions have INPUT policy configured"""
    bf.asserts.current_assertion = 'Assert all customer BGP sessions have input route policy'

    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)
```

INTENTIONET

DEMO #2 – Test a routing policy change



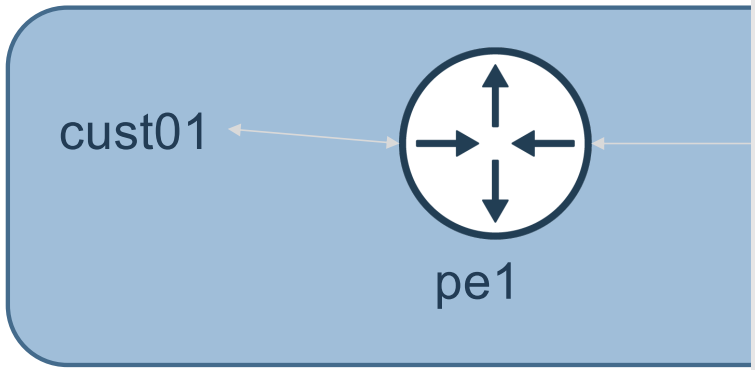
DEMO #2 – Test a routing policy change



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

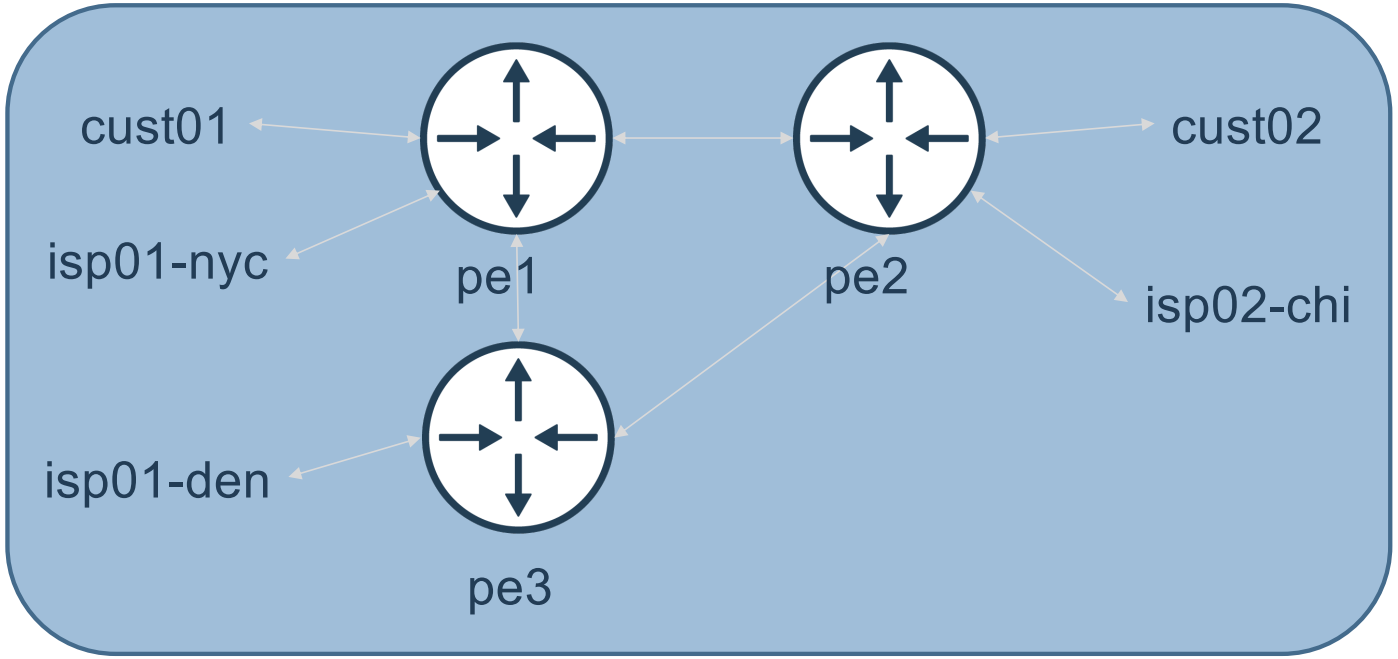
```
ip as-path access-list customer1 permit ^609
!
ip prefix-list customer1 deny 10.0.0.0/8 le 32
ip prefix-list customer1 deny 0.0.0.0/0
ip prefix-list customer1 deny 127.0.0.0/8 le 32
ip prefix-list customer1 deny 0.0.0.0/8 le 32
ip prefix-list customer1 permit 0.0.0.0/0 le 32
!
ip prefix-list BLOCK24 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

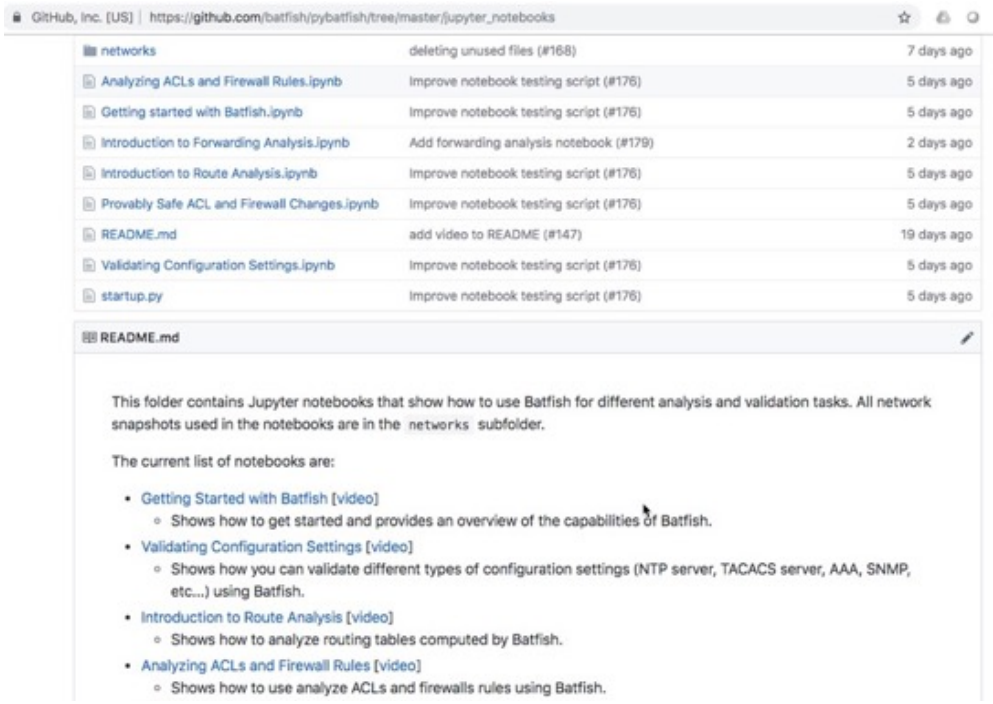
Getting Started with Batfish is easy!

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.youtube.com/channel/UCA-OUW_3IOt9U_s60KvmJYA



Download the presentation and demo

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

Validate your BGP policies with Batfish!

batfish-org

@batfish

batfish.org


Thank You
INTENTIONET
YOUR NETWORK. EXACTLY AS YOU INTENDED.