Peer Risk Intelligence

Should You Peer With That ASN?

[Your Name]

NANOG [XX] | [Date]

The Problem

"Is this network safe to peer with?"

- Peering decisions based on **PeeringDB + gut feel**
- No visibility into routing behavior before peering
- Discover problems *after* the BGP session is up
- Route leaks and hijacks from peers are expensive

Real Example: Cloudflare 1.1.1.1 (June 2024)

- AS267613 (Eletronet) announced 1.1.1.1/32
- AS262504 leaked 1.1.1.0/24 to upstreams
- Affected traffic for **7+ hours**
- Would you have peered with AS267613?

> **Question:** Could we have known this network was risky *before* the incident?

Introducing Route Sherlock

Open-source CLI tool for **Peer Risk Intelligence**

Data Sources

- RIPEstat (BGP data)
- PeeringDB (network metadata)
- BGPStream (historical archives)
- Claude API (AI synthesis)

Key Features

- Peer risk scoring
- Historical backtesting
- IX overlap analysis
- AI-powered reports

The Command

```
$ route-sherlock peer-risk AS64500 # Options: $ route-sherlock peer-risk
AS64500 --my-asn AS13335 # IX overlap $ route-sherlock peer-risk AS64500 --
days 180 # Extended history $ route-sherlock peer-risk AS64500 --ai # AI
analysis
```

Demo: Scoring Cloudflare (AS13335)

```
$ route-sherlock peer-risk AS13335 ╔═══════════════════════════ Peer Risk Score
═══════════════════════════╗ ╟ 100/100 (100.0%) ╟ ╟ Risk Level: LOW ╟ ╟
Recommendation: RECOMMENDED ╟
╚═══════════════════════════════════════════════════════════════════════════╝ Network:
CLOUDFLARENET | Type: Content | Policy: Open | IXes: 350
```

✓ Perfect score - safe to peer

Demo: Scoring Eletronet (AS267613)

```
$ route-sherlock peer-risk AS267613 ╔═══════════════════════════ Peer Risk
Score ═══════════════════════════╗ ╟ 72/100 (72.0%) ╢ ╟ Risk Level: MODERATE ╢
╟ Recommendation: ACCEPTABLE WITH MONITORING ╢
╚═══════════════════════════════════════════════════════════════════════════╝
Stability: 5/30 - High churn: 1637 updates/day (-25) ⚠️ Warning: High BGP
churn detected
```

⚠ Flagged BEFORE the incident occurred

Scoring Algorithm (100 points)

Category	Max	What We Check
Maturity	**20**	PeeringDB presence, IRR as-set, policy URL, IX count
Stability	**30**	BGP update frequency (churn detection)
Incident History	**30**	Upstream diversity, topology redundancy
Policy	**10**	Open (+10), Selective (+7), Restrictive (+3)
Security	**10**	IRR registration, transit relationships

Risk Levels & Recommendations

Score	Risk Level	Recommendation
80-100	LOW	Recommended - standard peering process
60-79	MODERATE	Acceptable - implement monitoring
40-59	ELEVATED	Caution - strict prefix limits, IRR filtering
0-39	HIGH	Not recommended - decline or require remediation

Stability Score: BGP Churn Detection

How it works

- Query RIPEstat for BGP updates
- Calculate updates per day
- Deduct points for high churn

Thresholds:
<10/day = Stable (no penalty)
10-50/day = Some activity
50-100/day = Moderate churn
>100/day = High churn (major penalty)

Real Results

ASN	Updates/Day	Score
AS13335	Low	30/30
AS15169	Low	30/30
AS267613	1,637	5/30

IX Overlap Analysis

```
$ route-sherlock peer-risk AS15169 --my-asn AS13335 ## IX Overlap Common
IXes: 164 Your IXes: 350 | Target IXes: 198 ✓ Can peer at 164 location(s)
```

- Instantly see where you can peer
- No need to manually compare PeeringDB
- Useful for peering coordinators

Historical Backtesting

```
$ route-sherlock backtest 1.1.1.0/24 --origin AS13335 \ --time "2024-06-27
18:00" --duration 8h 🚨 Anomalies Detected: 329 #1 [HIGH] ROUTE LEAK Time:
2024-06-27T18:49:06 AS Path: 50763 → 1031 → 262504 → 267613 → 13335 📅
Timeline: First anomaly: 18:49:06 UTC Duration: 7.6 hours
```

Validation: Did It Work?

Cloudflare's Report

- Incident started: ~18:51 UTC
- AS267613 announced 1.1.1.1/32
- AS262504 leaked to upstreams
- Duration: ~7.5 hours

Route Sherlock Found

- First anomaly: **18:49:06 UTC**
- Identified AS267613 ✓
- Identified AS262504 ✓
- Duration: 7.6 hours ✓

> ✓ **Detected 2 minutes before** Cloudflare's reported start time

AI-Powered Analysis (Optional)

```
$ route-sherlock peer-risk AS267613 --ai ┌──────────────────── AI-Generated Risk
Assessment ────────────────────┐ | | | **Executive Summary** | | Conditional peer.
AS267613 shows concerning stability metrics | | with 1,637 BGP updates/day. If peering
is necessary, implement | | strict safeguards. | | | | **Technical Safeguards** | | •
Max-prefix limit: 15 (they announce 7 prefixes) | | • Require IRR filtering against
RADB::AS-267613 | | • Enable RPKI-invalid rejection | | • Configure BGP session alerting
for prefix changes | | |
└──────────────────────────────────────────────────────────────────────────────┘
```

How Is This Different?

Feature	BGPalerter	ARTEMIS	Radar	Route Sherlock
Real-time monitoring	✓	✓	✓	✗
Historical backtesting	✗	✗	Limited	✓
Peer risk scoring	✗	✗	✗	✓
AI analysis	✗	✗	✗	✓
"Should I peer?"	✗	✗	✗	✓
Open source	✓	✓	✗	✓

Key insight: BGPalerter monitors YOUR prefixes. Route Sherlock evaluates OTHER networks before you peer.

Practical Safeguards by Risk Level

Risk	Max-Prefix	IRR Filter	RPKI	Monitoring
LOW	2x announced	Standard	Warn on invalid	Standard
MODERATE	1.5x announced	Strict	Reject invalid	Alert on changes
ELEVATED	1.2x announced	Strict + verify	Reject invalid	Alert + review
HIGH	Decline or require remediation first			

Tool provides specific recommendations based on their actual prefix count and IRR registration.

Getting Started

```
# Install $ pip install route-sherlock # For historical backtesting $ brew
install bgpstream $ pip install pybgpstream # Optional: AI synthesis $
export ANTHROPIC_API_KEY="your-key" # Run $ route-sherlock peer-risk AS64500
```

Summary

- **Problem:** No tool answers "Should I peer with this ASN?"
- **Solution:** Route Sherlock - Peer Risk Intelligence
- **How:** Combines RIPEstat + PeeringDB + BGPStream + AI
- **Validated:** Correctly flags networks involved in real incidents
- **Practical:** Outputs actionable recommendations

> **Try it:** Score a network you're considering peering with today

Questions?

GitHub: [your-repo-url]

[your-email]

[your-twitter/linkedin]

Backup: Architecture

```
route_sherlock/ ├── cli/ | ├── main.py # Typer CLI entry point | └── commands.py # Command implementations ├── collectors/ | ├── ripestat.py # RIPEstat API client | ├── peeringdb.py # PeeringDB API client | └── bgpstream.py # Historical BGP archives └── synthesis/ └── engine.py # AI synthesis with Claude
```

- Python 3.11+ with async/await
- Rich for terminal UI
- pybgpstream for RouteViews/RIPE RIS