bwNetFlow:
A Customizable Multi-Tenant Flow Processing Platform for Transit Providers

Daniel Nägele, Christopher B. Hauser, Leonard Bradatsch, Stefan Wesner
SC19: INDIS Workshop, 11/17/2019
Introduction

- Daniel Nägele (naegele@belwue.de)
- Researcher **bwNetFlow** project

- Working at AS553 (BelWü)
  - Regional research and education network
  - Serving 9 universities, 46 colleges, among others
  - Aggregate transit capacity of ~1 Tbit/s
  - A lot of peering
Assumptions and Goals

- Monitor traffic on all border interfaces
- Researchers have challenging flexibility requirements
  - Treat flows as discrete messages for maximum flexibility
  - Provide interested parties with solely their specific flows to...

enable operative and scientific insights
enhance applications with live data
visualize using simple dashboards
access the full data using an API
Apache Kafka as a core element for bwNetFlow

- **Apache Kafka**\(^1\) is a distributed streaming platform
- *Topics* are ordered streams of *protobuf*-encoded\(^2\) flow objects
- Topics are consumed and produced
- Built-in support for...
  - encryption
  - load balancing
  - retention policies
  - access control
  - partitioning
  - replication
- **goflow**\(^3\) is a Netflow Collector for Apache Kafka

---

1. kafka.apache.org
2. developers.google.com/protocol-buffers
3. github.com/cloudflare/goflow
Apache Kafka as a core element for bwNetFlow

- **Apache Kafka**\(^1\) is a distributed streaming platform
- *Topics* are ordered streams of *protobuf*-encoded\(^2\) flow objects
- Topics are *consumed* and *produced*
- Built-in support for ...
  - encryption
  - load balancing
  - retention policies
  - access control
  - partitioning
  - replication

- **goflow**\(^3\) is a Netflow Collector for Apache Kafka

---

\(^1\)kafka.apache.org
\(^2\)developers.google.com/protocol-buffers
\(^3\)github.com/cloudflare/goflow
Project Architecture with Kafka at its core

- protobuf format
- extensible and efficient
Project Architecture with Kafka at its core

- Kafka Cluster
  - ingess topic
  - enriched topic

- Producers
- Topics
- Processors
- Enricher

- Routers

- goflow
- fra-decix-1
- fra-decix-2

- Our Customers
- Custom Consumers (Go, C, Python, Java, …)
- Dashboards
- Evaluations

- BelWü

- IP prefix (IPAM, CSV)
- remote geolocation
- interface data (SNMP)
- routing info
- arbitrary lookups

- Protobuf format
- extensible and efficient
- IP prefix (IPAM, CSV)
- remote geolocation
- interface data (SNMP)
- routing info
- arbitrary lookups

- InfluxDB
- Prometheus
- CSV and nfdump

- Authenticated Access (e.g. through Grafana)
  - protobuf format
  - extensible and efficient
  - IP prefix (IPAM, CSV)
  - remote geolocation
  - interface data (SNMP)
  - routing info
  - arbitrary lookups
Project Architecture with Kafka at its core

Kafka Cluster

- ingress topic
- enriched topic
- customer topics

Routers
- fra-decix-1
- fra-decix-2
- goflow

Producers
- Topics

Processors
- Enricher
- Splitter

Consumers
- Various formats

- BelWü
- Our Customers
- Custom Consumers (Go, C, Python, Java, …)
- Dashboards
- Evaluations
- Netflow v9
- Authenticated Access (e.g. through Grafana)
  - protobuf format
  - extensible and efficient
  - IP prefix (IPAM, CSV)
  - remote geolocation
  - interface data (SNMP)
  - routing info
  - arbitrary lookups
  - split on any field
  - supports filters

- InfluxDB
- Prometheus
- CSV and nfdump
Project Architecture with Kafka at its core

- Routers
  - fra-decix-1
  - fra-decix-2

- Kafka Cluster
  - ingress topic
  - enriched topic
  - customer topics
  - minimized topics

- Processors
  - Enricher
  - Splitter
  - Reducer

- Consumers
  - goflow
  - BelWü
  - Custom Consumers (Go, C, Python, Java, …)
  - Dashboards

- Evaluations
  - Authenticated Access (e.g. through Grafana)
  - Netflow v9
  - protobuf format
  - extensible and efficient
  - IP prefix (IPAM, CSV)
  - remote geolocation
  - interface data (SNMP)
  - routing info
  - arbitrary lookups
  - split on any field
  - supports filters
  - remove fields
  - drop flows
  - anonymization

- Protocols
  - InfluxDB
  - Prometheus
  - CSV and nfdump
Project Architecture with Kafka at its core

Kafka Cluster

- Routers
  - fra-decix-1
  - fra-decix-2

- goflow

- Topics
  - ingress topic
  - enriched topic
  - customer topics
  - minimized topics

- Processors
  - Enricher
  - Splitter
  - Reducer

- Consumers
  - various formats

- InfluxDB
- Prometheus
- CSV and nfdump

- BelWü
- Our Customers
- Custom Consumers (Go, C, Python, Java, ...)
- Dashboards
- Evaluations

- Authenticated Access (e.g. through Grafana)
  - protobuf format
  - extensible and efficient
  - IP prefix (IPAM, CSV)
  - remote geolocation
  - interface data (SNMP)
  - routing info
  - arbitrary lookups
  - split on any field
  - supports filters
  - remove fields
  - drop flows
  - anonymization

  - InfluxDB
  - Prometheus
  - CSV and nfdump

8
Project Architecture with Kafka at its core

Kafka Cluster

- ingress topic
- enriched topic
- customer topics
- minimized topics
- various formats

BelWü

- goflow
- fra-decix-1
- fra-decix-2

Our Customers

- Official Application
  - Dashboards
  - Evaluations
  - Netflow v9

- Custom Consumers
  (Go, C, Python, Java, ...)

- Authenticated Access
  (e.g. through Grafana)

- Evaluations
- Netflow v9

- Protocols
  - protobuf format
  - extensible and efficient
  - IP prefix (IPAM, CSV)
  - remote geolocation
  - interface data (SNMP)
  - routing info
  - arbitrary lookups
  - split on any field
  - supports filters
  - remove fields
  - drop flows
  - anonymization
  - InfluxDB
  - Prometheus
  - CSV and nfdump

Routers

- Producers
- Topics
- Processors
- Enricher
- Splitter
- Reducer
- Consumers
Interactive Visualizations
Interactive Visualizations: Geolocation
from confluent_kafka import Consumer
import flow_messages_enriched_pb2 as api

consumer = Consumer(config)  # static config (host, ssl, sasl authentication)
consumer.subscribe(['flow-messages-enriched'])

while True:
    # Step 1: get data from kafka cluster
    raw = consumer.poll()

    # Step 2: decode using google protobuf
    flow = api.FlowMessage().ParseFromString(raw.value())

    # Step 3: work with the flow
    pass
Examples from our ops team’s Git

Which peers should fix some ACLs?

dst = ipaddress.ip_address(flow.DstAddr)
if not dst.is_global:
  print(flow.Peer)

Who has hosts talking to known Command & Control servers?

badguy = bytes([81, 169, 145, 160]):
if badguy in (flow.SrcAddr, flow.DstAddr):
  print(f"{flow.Cid}: {flow.SrcAddr} -> {flow.DstAddr}"")

Where do my users access my site from?

# distribution is a defaultdict: {'DE': 100, 'US': 70, ...}
if flow.DstAddr == bytes([129, 143, 232, 10]):
Examples from our ops team’s Git

Which peers should fix some ACLs?

dst = ipaddress.ip_address(flow.DstAddr)
if not dst.is_global:
    print(flow.Peer)

Who has hosts talking to known Command & Control servers?

badguy = bytes([81,169,145,160]):
if badguy in (flow.SrcAddr, flow.DstAddr):
    print(f"{flow.Cid}: {flow.SrcAddr} -> {flow.DstAddr}"

Where do my users access my site from?

# distribution is a defaultdict: {'DE': 100, 'US': 70, ...}
if flow.DstAddr == bytes([129,143,232,10]):
Examples from our ops team’s Git

Which peers should fix some ACLs?

dst = ipaddress.ip_address(flow.DstAddr)
if not dst.is_global:
    print(flowPeer)

Who has hosts talking to known Command & Control servers?

badguy = bytes([81,169,145,160]):
if badguy in (flow.SrcAddr, flow.DstAddr):
    print(f"{flow.Cid}: {flow.SrcAddr} -> {flow.DstAddr}"

Where do my users access my site from?

# distribution is a defaultdict: {'DE': 100, 'US': 70, ...}
if flow.DstAddr == bytes([129,143,232,10]):
Applications developed by our Customers: CLI Tools

[danieln@waystone ~]$ flowtop

<table>
<thead>
<tr>
<th>Totals</th>
<th>Remote Location</th>
<th>AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits: 56.4831 Gbps</td>
<td>US: 43.2% - 24.428 Gbps</td>
<td>other: 79.7% - 4.504 Gbps</td>
</tr>
<tr>
<td>Packets: 8898528/s</td>
<td>DE: 27.2% - 15.376 Gbps</td>
<td>netflix: 5.9% - 3.385 Mbps</td>
</tr>
<tr>
<td>Flows: 43610.40/s</td>
<td>IE: 9.9% - 5.609 Gbps</td>
<td>amazon: 4.7% - 2.702 Mbps</td>
</tr>
<tr>
<td></td>
<td>NL: 5.1% - 2.927 Gbps</td>
<td>dtag: 2.9% - 1.672 Mbps</td>
</tr>
<tr>
<td></td>
<td>GB: 2.8% - 1.583 Gbps</td>
<td>steam: 1.3% - 756.1 Mbps</td>
</tr>
<tr>
<td></td>
<td>CA: 2.5% - 1.457 Gbps</td>
<td>msoft: 1.3% - 753.0 Mbps</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address Family</th>
<th>Protocols</th>
<th>Peers</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6: 5.07% - 2861.6 Mbps</td>
<td>TCP: 86.6% - 48.94 Gbps</td>
<td>DE-CIX: 18.5% - 1.046 Gbps</td>
</tr>
<tr>
<td>othe: 0.00% - 0.00 Mbps</td>
<td>othe: 1.6% - 933.3 Mbps</td>
<td>ECIX: 16.3% - 9.219 Mbps</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Direction</th>
<th>Protocols</th>
<th>Peers</th>
</tr>
</thead>
<tbody>
<tr>
<td>In: 73.86% - 41.717 Gbps</td>
<td>TCP: 86.6% - 48.94 Gbps</td>
<td>DE-CIX: 18.5% - 1.046 Gbps</td>
</tr>
<tr>
<td>N/A: 0.00% - 0.00 Mbps</td>
<td>othe: 1.6% - 933.3 Mbps</td>
<td>ECIX: 16.3% - 9.219 Mbps</td>
</tr>
</tbody>
</table>

>
Applications developed by our Customers: DDoS Detection

Currently no attack in this network.

source port entropy (measured) vs. destination IP entropy (measured)

source port entropy (predicted) vs. destination IP entropy (predicted)

all flows (measured) vs. TCP flows (measured) vs. UDP flows (measured)

all flows (predicted) vs. TCP flows (predicted) vs. UDP flows (predicted)

bwNet100G+ project, www.bwnet100g.de
Thomas Lukaseder, uulm.de/?seder
• Improve Open Source presence and documentation
• Allow customers to influence their pipelines directly, without manual intervention
• Follow-up project bwNet2020 is approved, integrating both projects
  • major themes: Network Function Virtualization, Service Function Chaining
  • bwNetFlow as central component for the monitoring aspect as well as a service
Thank you!
Questions?

or contact me later: naegele@belwue.de
our code: github.com/bwNetFlow