Less is More with Intelligent Packet Capture

RANDY CALDEJON

FLOCON 2020

Objectives

- Consider merits of streaming analytics
- Expose to advanced open source tools
- Encourage to experiment with OpenArgus

DragonFly

Streaming Analytics at the Edge

- Increase speed
- Reduce bandwidth
- Local Resources

DragonFly Design Goals

Machine Learning

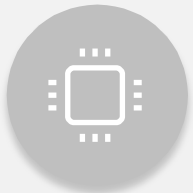
Analyzes data as it arrives

Incremental Updates

Receive updates before the flow is complete

Sustained Performance

Maintains 20Gbps+,



Single Node Architecture

High-performance without a cluster

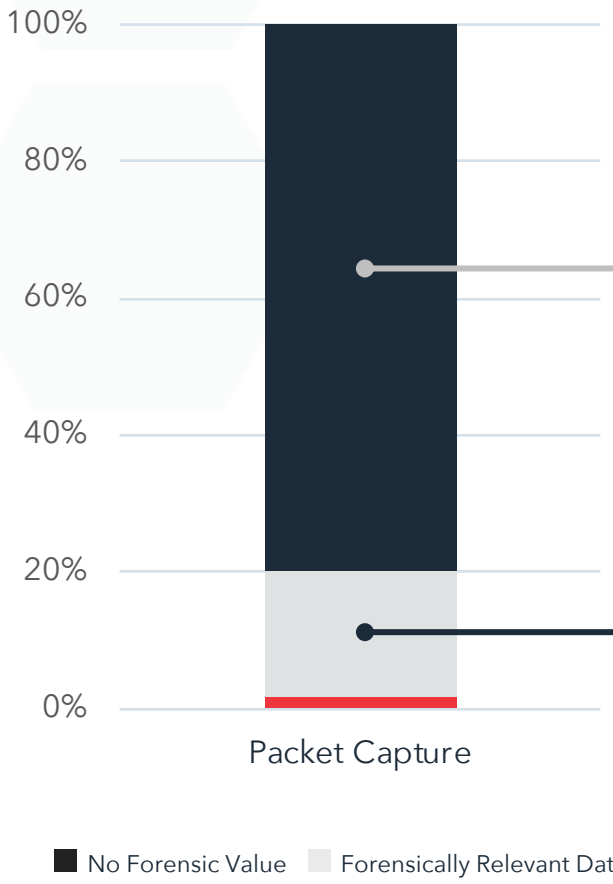
Bolt-On Mindset

Integrate seamlessly with other security tools

A Practical Application of DragonFly

PCAP or it didn't happen.

Full Packet Capture is Ground Truth; but...



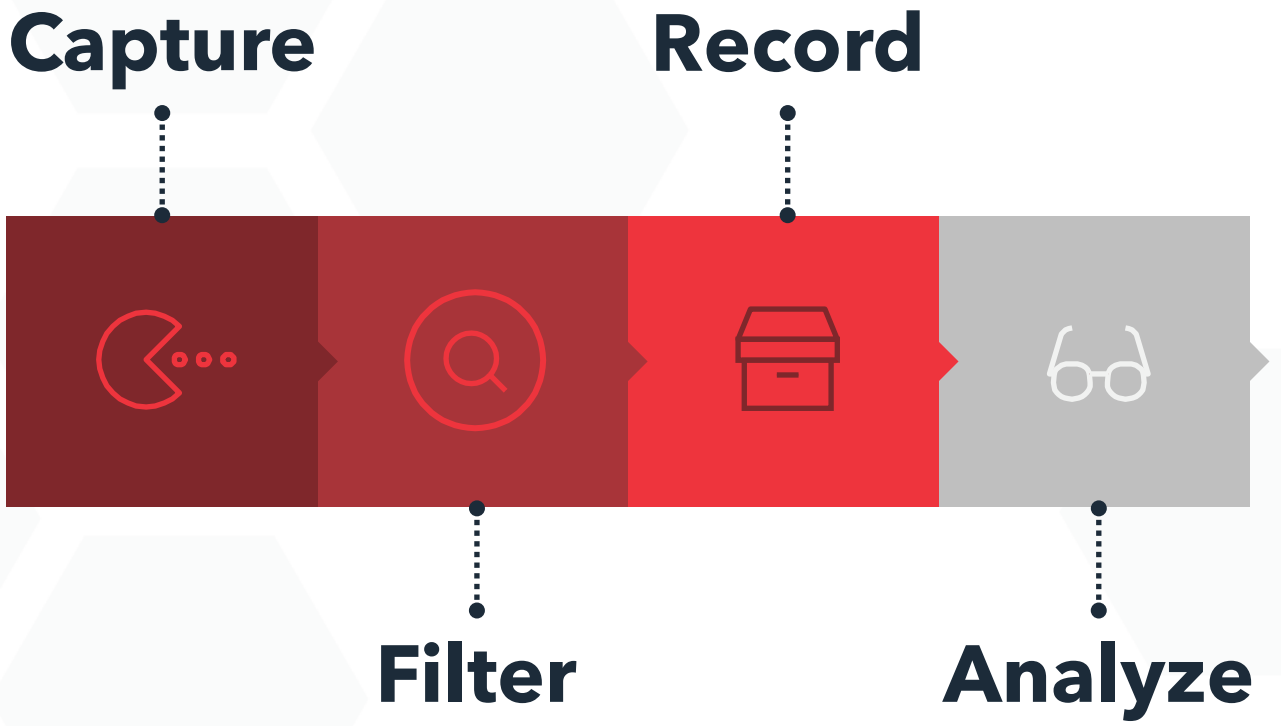
High Cost

10Gbps Network Link
30 days ***~$1.2M annually***

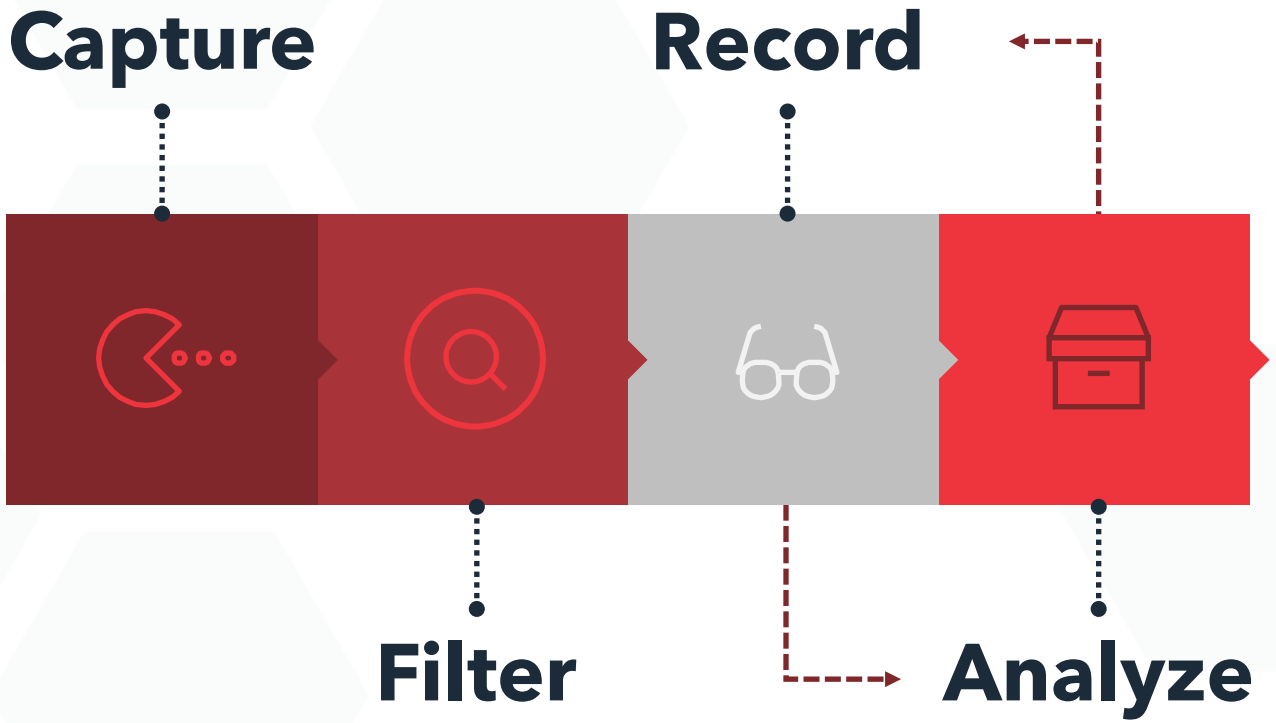
Low Signal to Noise

Forensically relevant network data is a small fraction of total network data

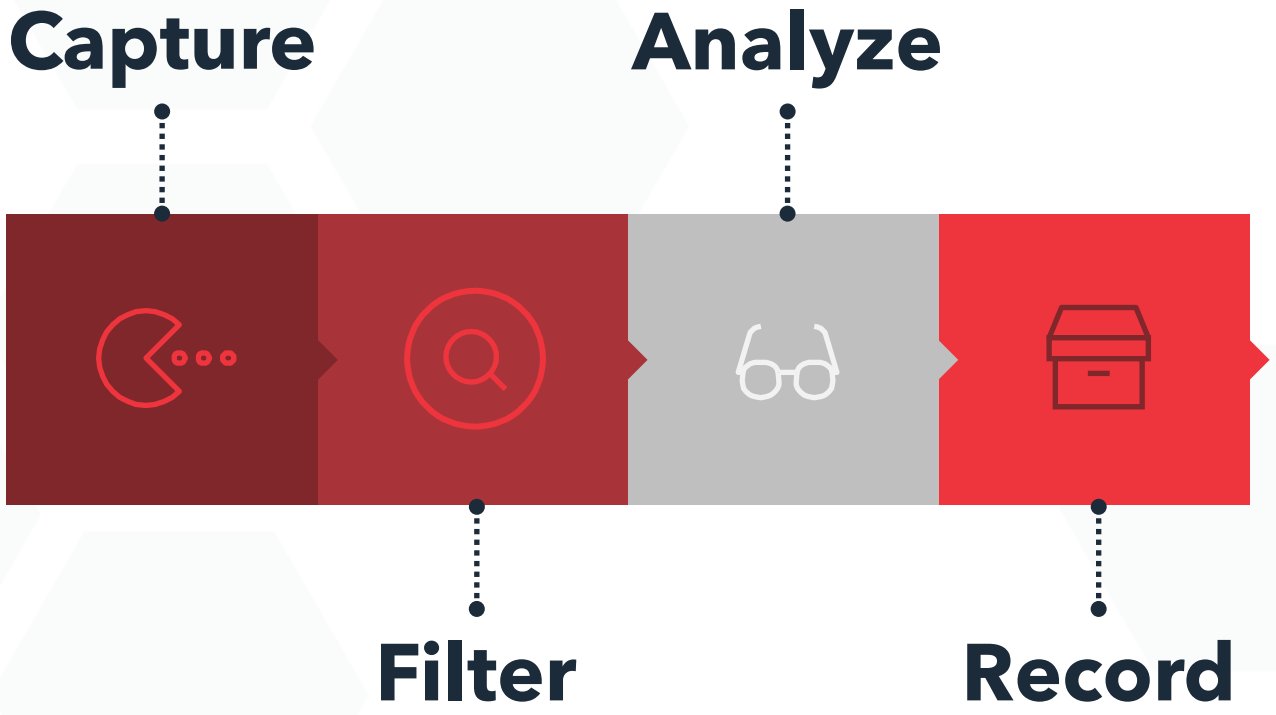
Typical Packet Capture Workflow: Retrospective



Intelligent Packet Capture



Intelligent Packet Capture: Real-Time



Intelligent PCAP

Using Machine Learning to Capture Packets with Forensic Value

Ground truth – Full packet capture has long been viewed as the "ground truth" for activity on the network, allowing analysts to identify the source of security incidents.

Expensive – Despite its value, full packet capture is not used to its fullest extent because lengthy retention periods are cost prohibitive and retention only shrinks as bandwidth utilization increases.

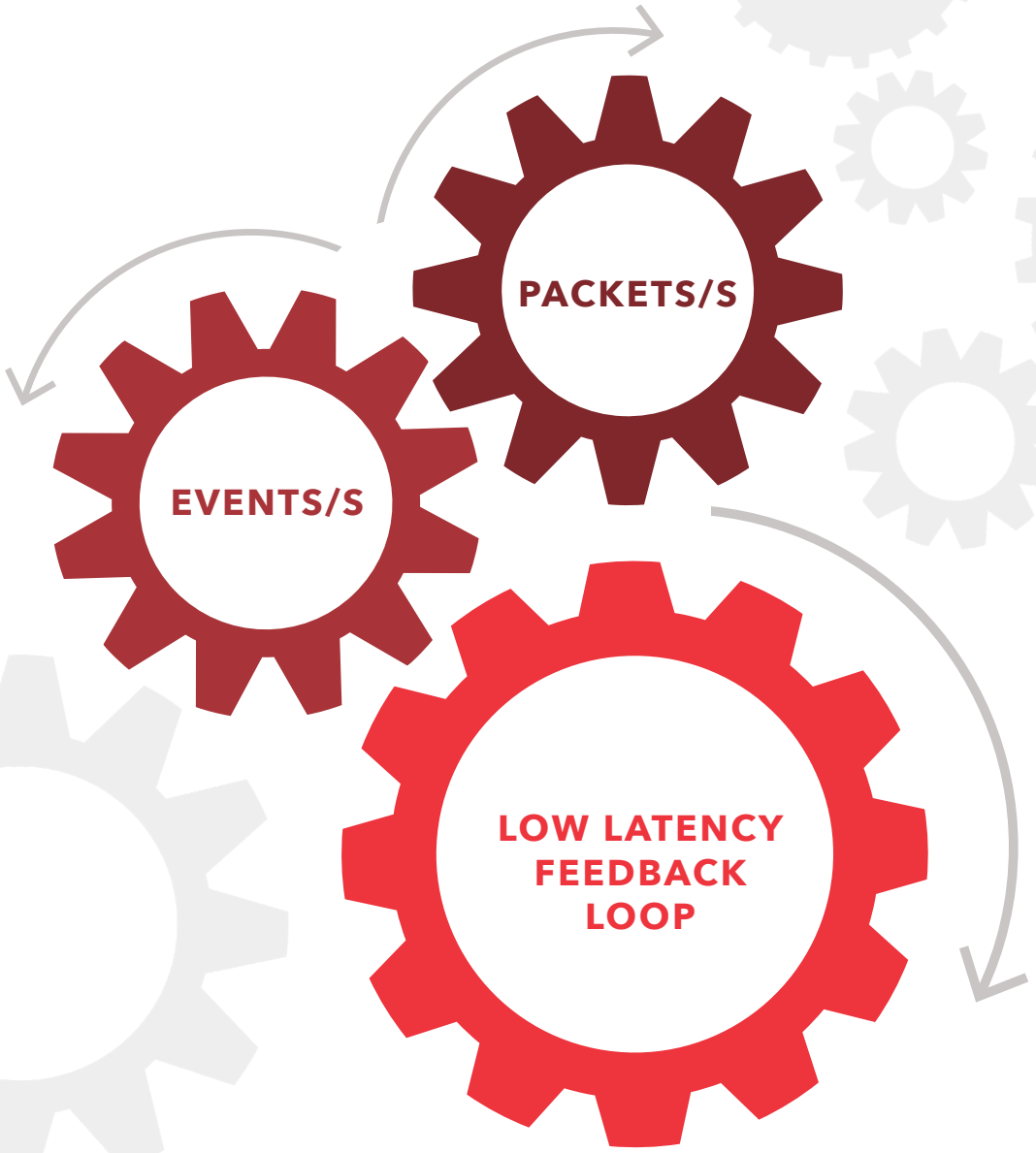
Alternatives Lack Payloads – Though valuable for portions of the security workflow, alternatives to PCAP such as Flow, and Application Metadata cannot provide the "ground truth" payload for irregular traffic.

Combine forces – Intelligent packet capture combined with augmented flow provides a powerful combination that supports a data friendly log format plus the full packets for anomalous traffic.

Intelligent Packet Capture

uses **threat intelligence**, **advanced analytics**, and **Machine Learning** to decide in near real-time what to record.

Intelligent PCAP
Performance Requirements



Intelligent PCAP

Open Source Framework

Argus
(extraction)

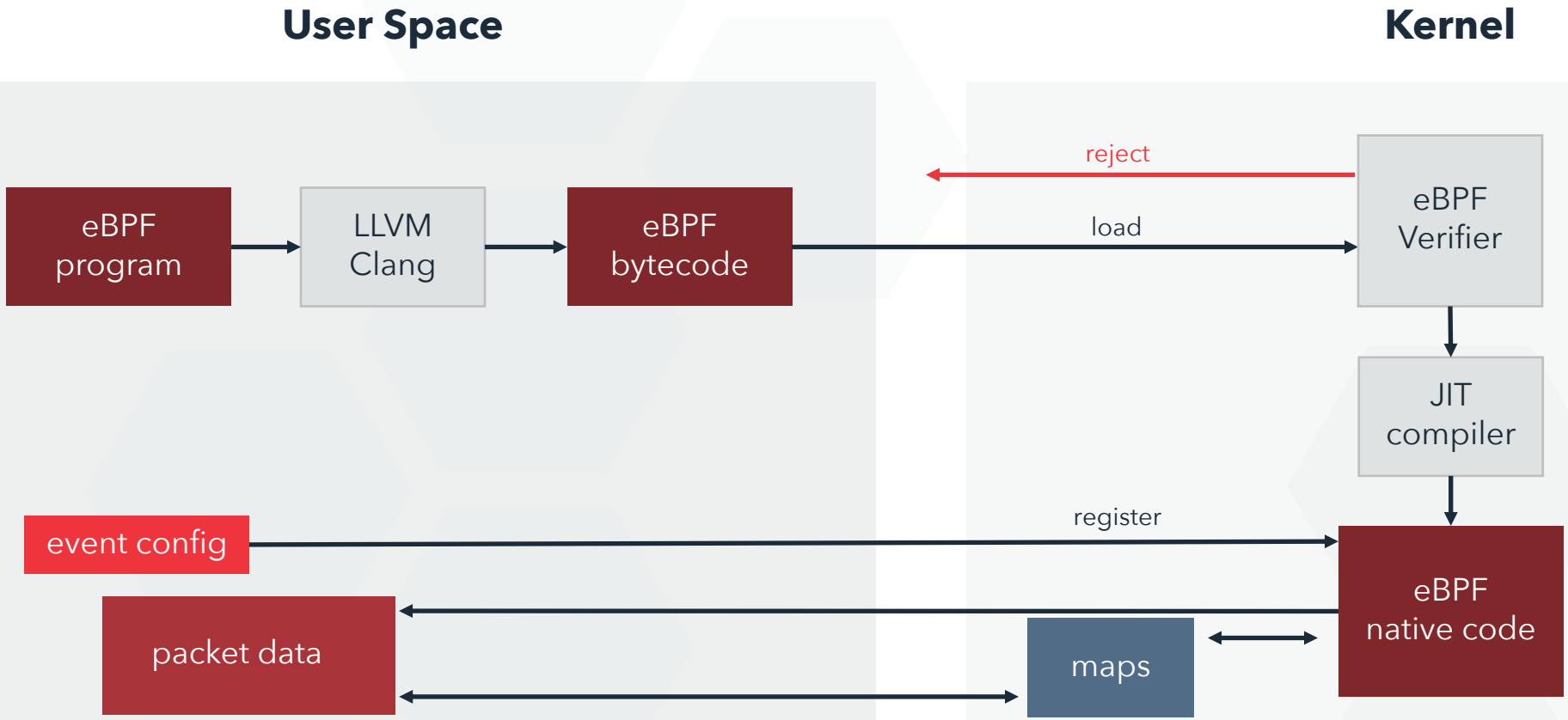
mlpack
(training)

eBPF
(filtering)

tcpdump
(recording)

```
tcpdump -i eth0 -w /cache/pcap-%m-%d-%H-%M-%S \
                -W 100 -G 300 -C 1000
```

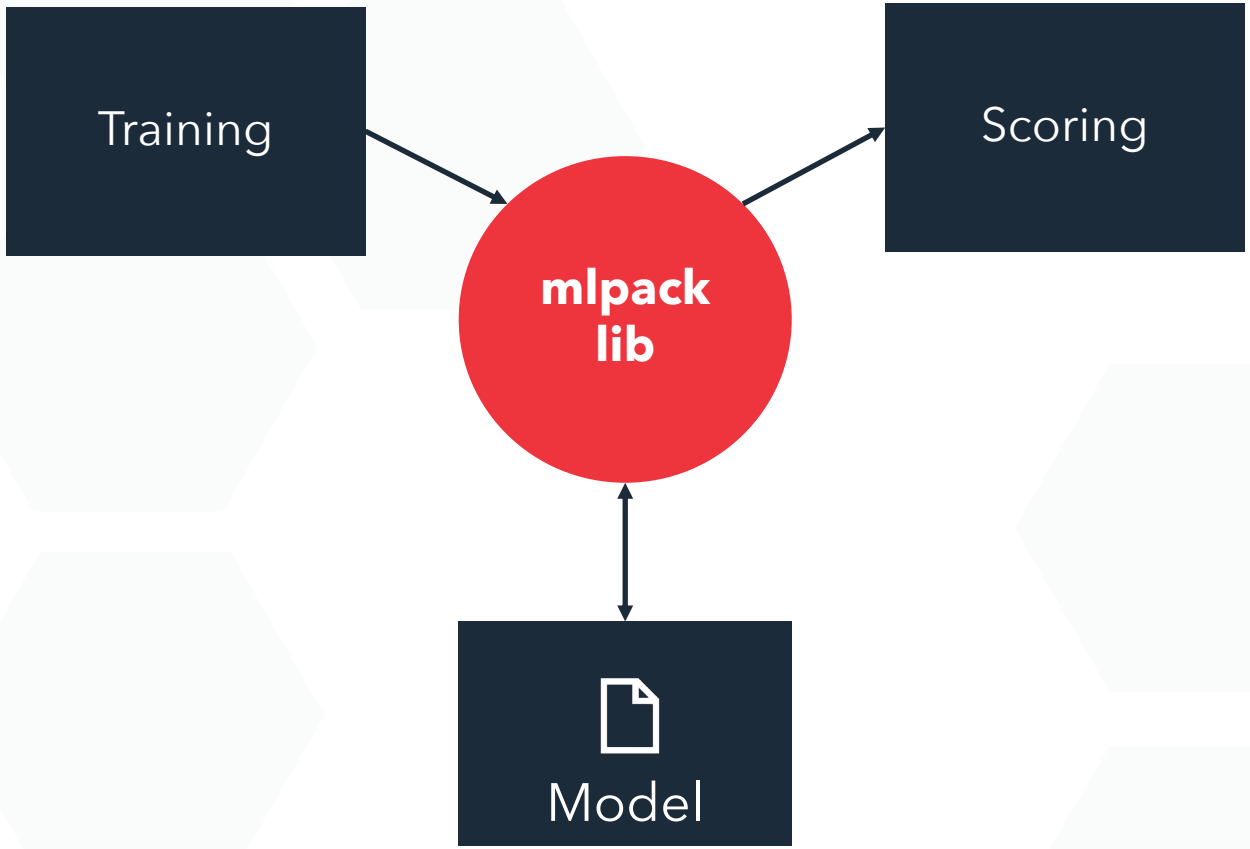
eBPF for Filtering



eBPF Map

```
struct bpf_map_def SEC("maps") watchlist = {

        .type         = BPF_MAP_TYPE_PERCPU_HASH,
        .key_size     = sizeof(u32), /* ipv4 address */
        .value_size   = sizeof(u64), /* counter/timeout */
        .max_entries = 100000,
        .map_flags    = BPF_F_NO_PREALLOC,

}
```


Mlpack for training



mlpack splitting data

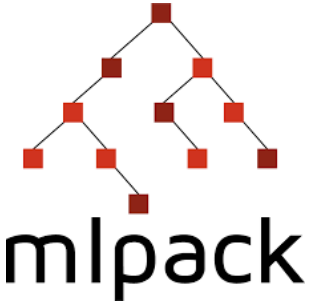
1

```
/usr/local/bin/mlpack_preprocess_split                          \
    --input_file data/$filename.data.csv                        \
    --input_labels_file data/$filename.labels.csv               \
    --training_file data/$filename.train.csv                    \
    --training_labels_file data/$filename.train.labels.csv      \
    --test_file data/$filename.test.csv                         \
    --test_labels_file data/$filename.test.labels.csv           \
    --test_ratio 0.3                                            \
    --verbose
```


mlpack generating model

2

```
/usr/local/bin/mlpack_random_forest                          \
    --training_file data/$filename.data.csv                  \
    --labels_file data/$filename.labels.csv                  \
    --num_trees 10                                           \
    --minimum_leaf_size 3                                    \
    --print_training_accuracy                                \
    --output_model_file model/$filename.eval-model.bin       \
    --verbose
```

mlpack testing model

3

```
/usr/local/bin/mlpack_random_forest                          \
    --input_model_file model/$filename.eval-model.bin \
    --test_file data/$filename.test.csv                      \
    --test_labels_file data/$filename.test.labels.csv \
    --probabilities_file probs.csv                           \
    --verbose
```

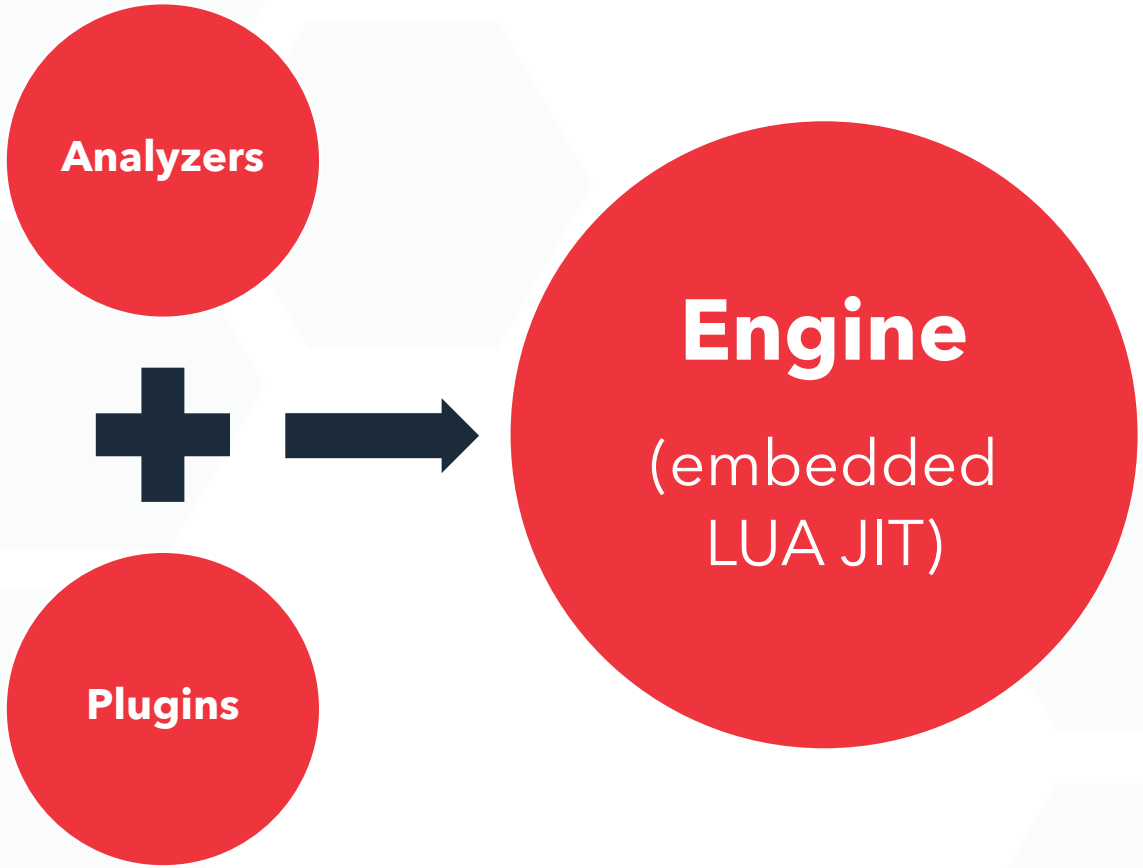

DragonFly

MACHINE LEARNING ENGINE

Version 2.0

- Scalable
- Lightweight
- Flexible
- Extensible

DragonFly MLE



DragonFly Engine

Fast - C/C++

Lightweight – Small Library

Scriptable – Embedded LUA JIT

Easy – Arduino Programming Model

DragonFly Scriptable Analyzers

```
function M:setup()
   model = config['module.model']
   rf = RandomForest.load(model)
end

function M:loop (event)
    ….
   rf:classify (event)
end
```

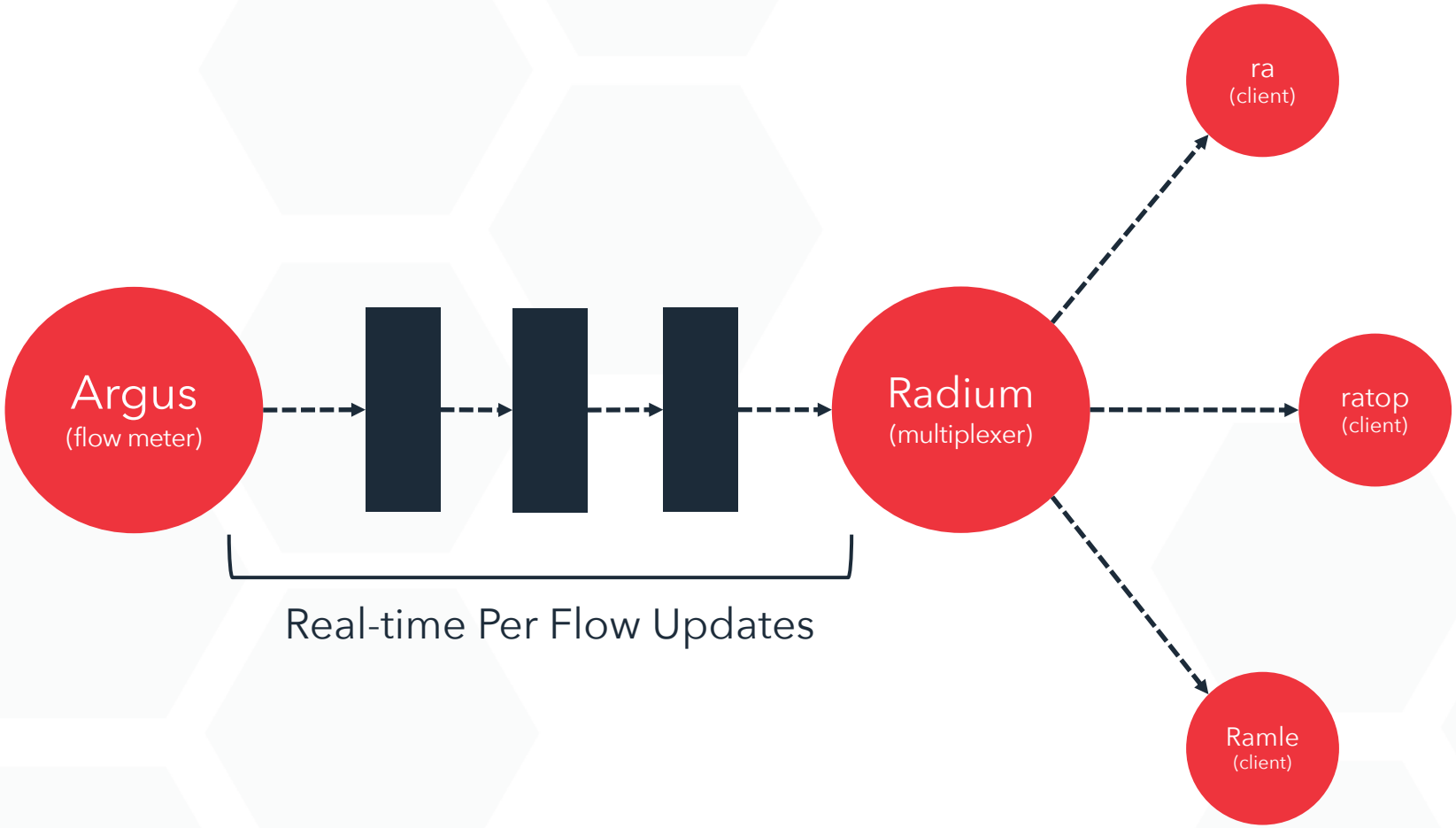

DragonFly Scriptable Analyzers

```
function M:dns (event)
    ….
    rf:classify (event)
end

function M:tls (event)
    ….
    rf:classify (event)
end
```


DragonFly Plug-ins

- mlpack
- eBPF
- iptree
- Redis
- cuckoo filter

Argus


Argus
(flow meter)
Radium
(multiplexer)
ra
(client)
ratop
(client)
Ramle
(client)
Real-time Per Flow Updates

argus

Argus

Real-Time Flow Meter

Flow Features

Packet Dynamic Features

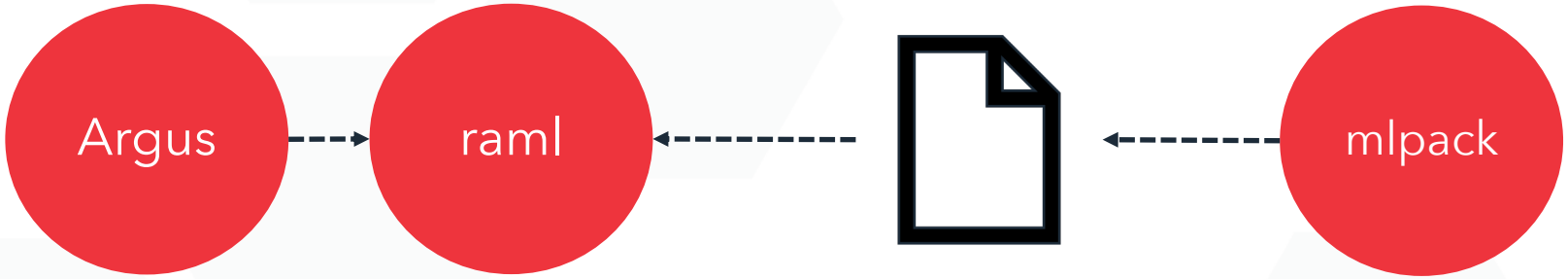
Field Overview			
Flow	• IP Addresses • Ports • Protocol	• Total Bytes • Total Packets	• Start time • Duration
Extended Flow	• Flow details by direction	• Payload	• MAC, VLAN, MPLS, ICMP, TCP flags and options
Packet Dynamics	• Connection Setup Times • Load and Rates (bytes and packets per second)	• Interpacket Arrival time and Jitter • Dropped/retransmitted packet statistics	• Connection statistics (FIN, RST, SYN, Window advertisements, Zero windows)
Computed Statistics	• Producer/Consumer Ratio • App/Byte Ratio	• Key Stroke Identification	• Flow Active Runtime Statistics
Derived Fields	• Country Code	• MAC Manufacturer (OUI)	
Record Management	• Record Cause (Start, Status, Stop, Close, Error)	• Unique Identifier (seq) • Sensor ID	• Record Type (“flow” or “management”)

100+ Features

Intelligent PCAP with raml

- Based on Argus client (library)
- Integrated with DragonFly (library)
- Able to run an instance per core

Intelligent PCAP with raml



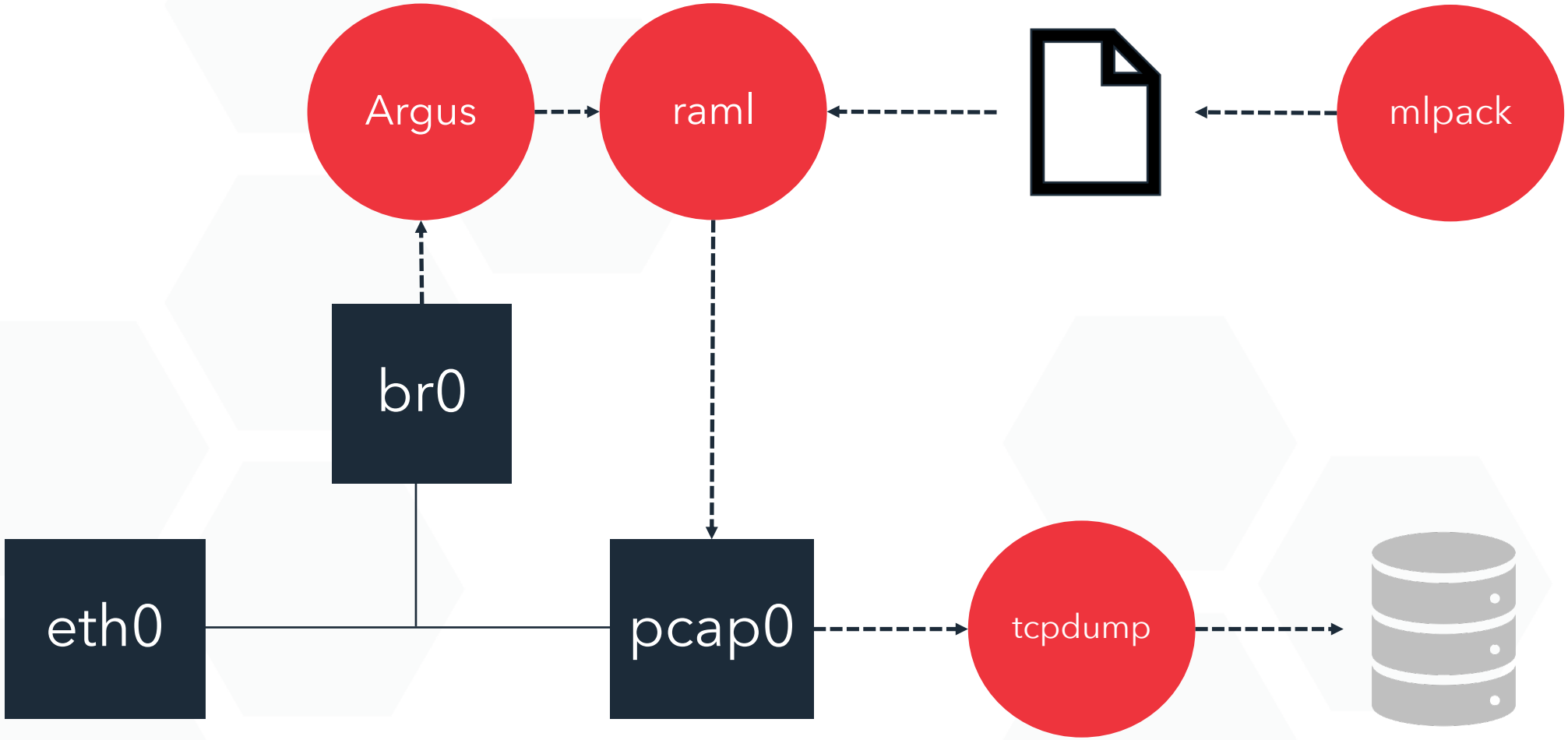
raml: DGA Analyzer

```
function M:loop (event)

    local v = features(event.domain,
event.ttl)
    score = rf:classify (v)

    return score
end
```


raml: Threat Feed Analyzer

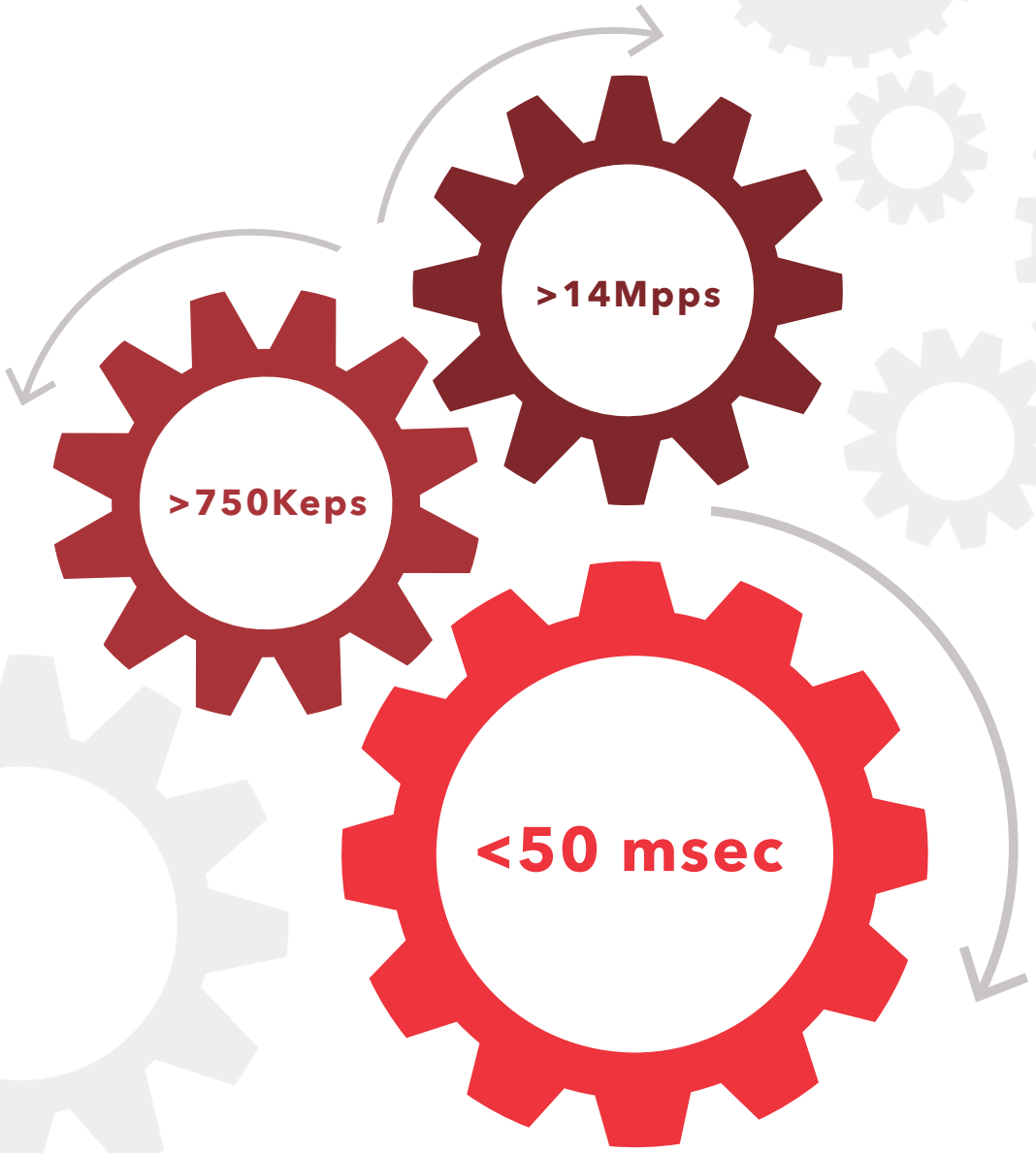
```
function M:setup()
    file = config['ioc.filename']
    iplist = iptree(file)
end

function M:loop (event)
    local daddr = event['daddr']
    match = iplist.lookup (daddr)
    return match
end
```


Intelligent PCAP Solutions



LESSONS LEARNED

Performance



Next Steps…

- **Complete POCs**
- **Publish to GitHub**
  https://github.com/counterflow-ai/dragonfly2
- **Merge raml with Argus**
  https://openargus.org/
- **Explore additional use cases…**

Streaming Analytics Use Cases

- Threat Intelligence Triage
- Encrypted Traffic Analysis
- Predictive Fault Detection

Questions?

RANDY CALDEJON

rc@counterflowai.com

https://github.com/counterflow-ai/dragonfly2