Pivot like a Pro: Unified Threat Hunting in Network Security Data

Matthias Vallentin

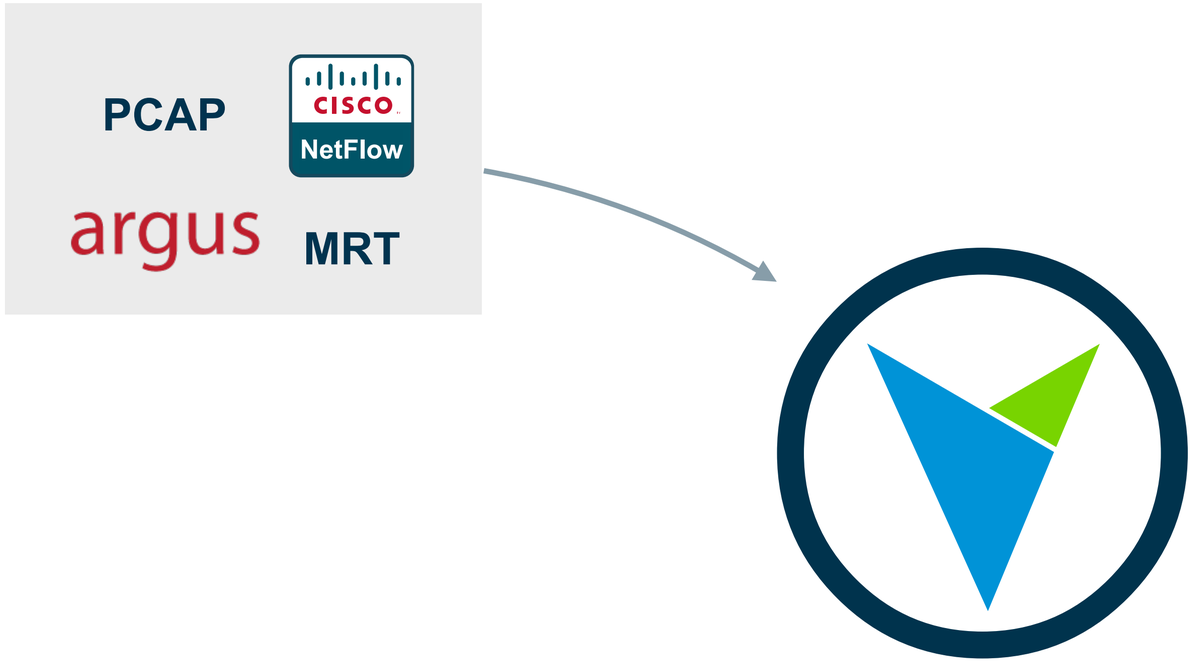
matthias@tenzir.com

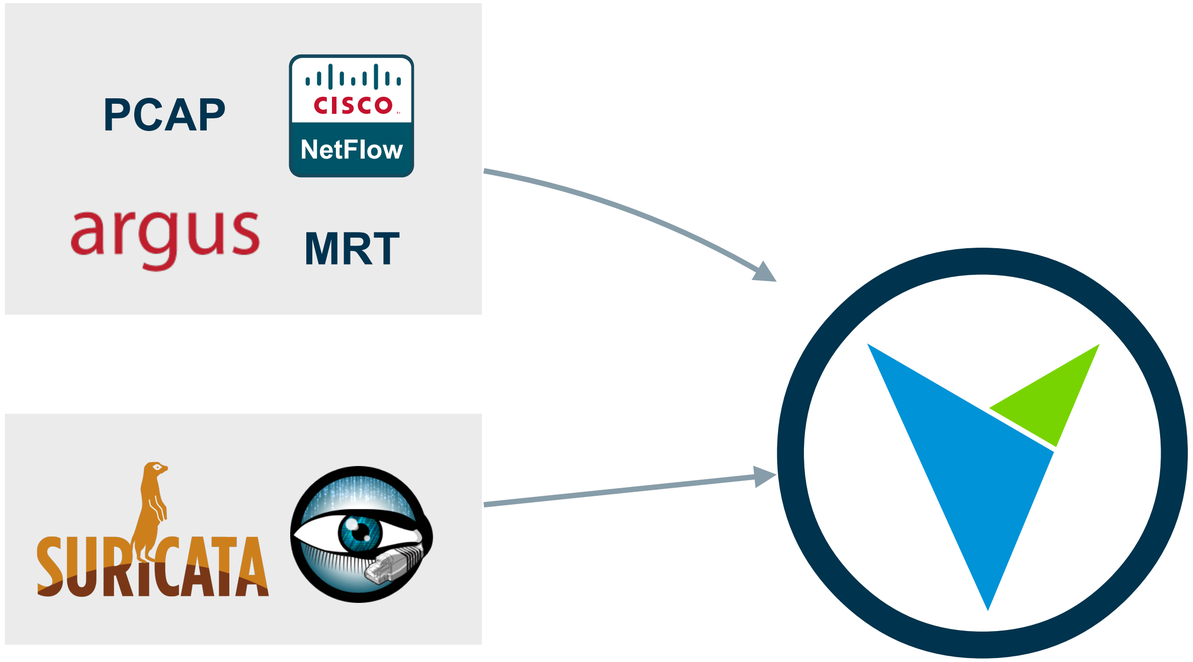
Suricon
October 30, 2019

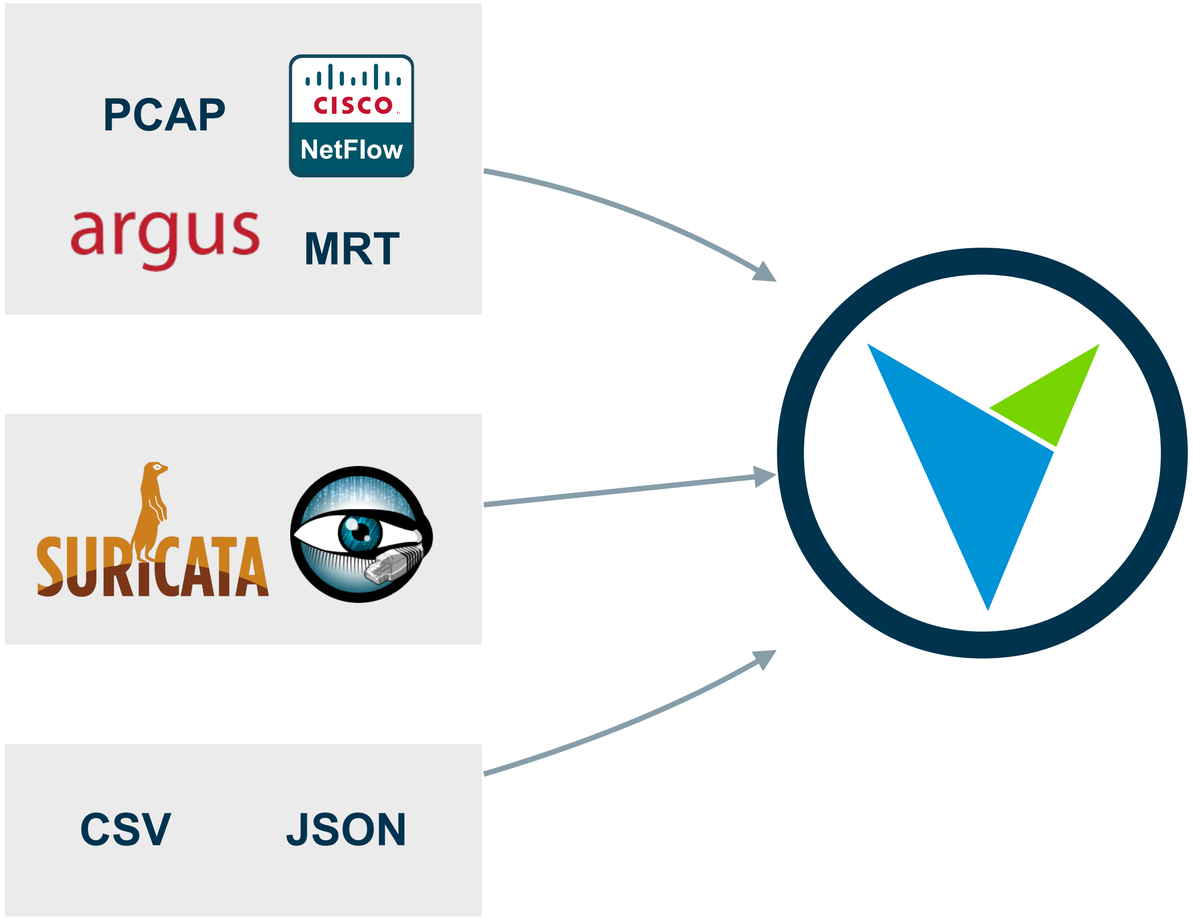
TENZIR

VAST

Visibility Across Space and Time

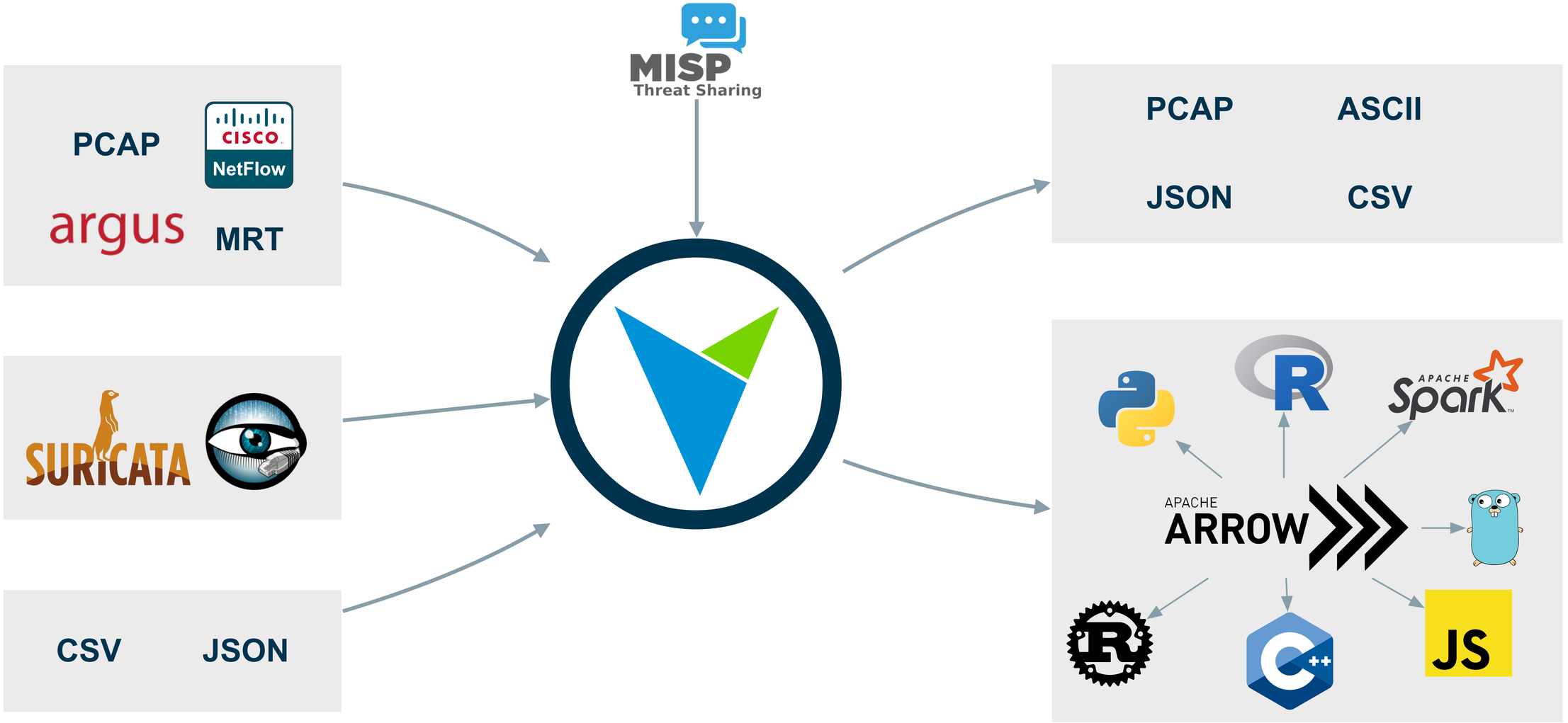

PCAP
CISCO
NetFlow
argus
MRT

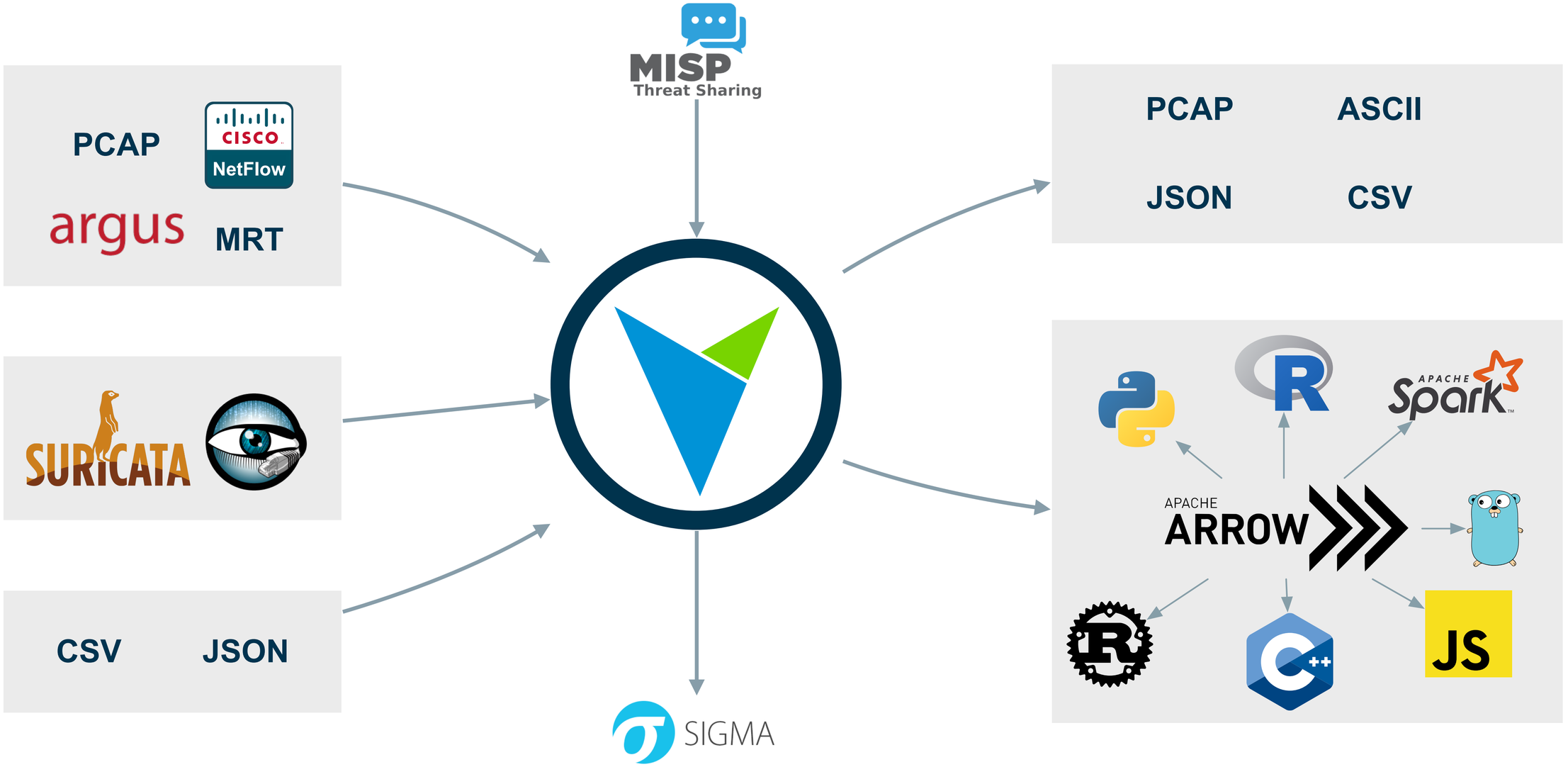

PCAP
CISCO
NetFlow
argus
MRT
SURICATA


PCAP
CISCO
NetFlow
argus
MRT
SURICATA
CSV
JSON

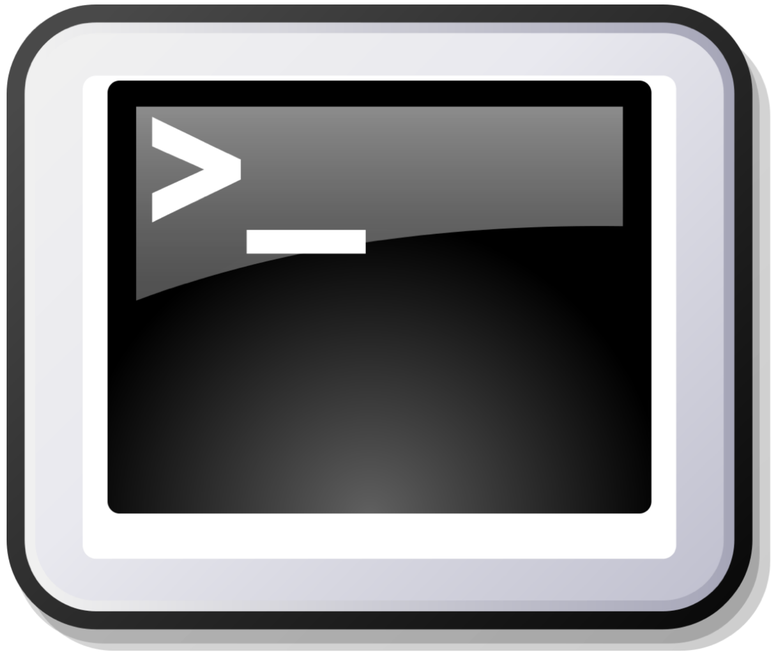

PCAP
NetFlow
argus
MRT
SURICATA
CSV
JSON
PCAP
ASCII
JSON
CSV


PCAP
NetFlow
argus
MRT
SURICATA
CSV
JSON
PCAP
ASCII
JSON
CSV
APACHE
Spark
APACHE
ARROW
JS


PCAP
NetFlow
argus
MRT
SURICATA
CSV
JSON
MISP
Threat Sharing
PCAP
ASCII
JSON
CSV
APACHE
Spark
APACHE
ARROW
JS


PCAP
CISCO
NetFlow
argus
MRT
SURICATA
CSV
JSON
MISP
Threat Sharing
SIGMA
PCAP
ASCII
JSON
CSV
APACHE
Spark
APACHE
ARROW
JS

Demo



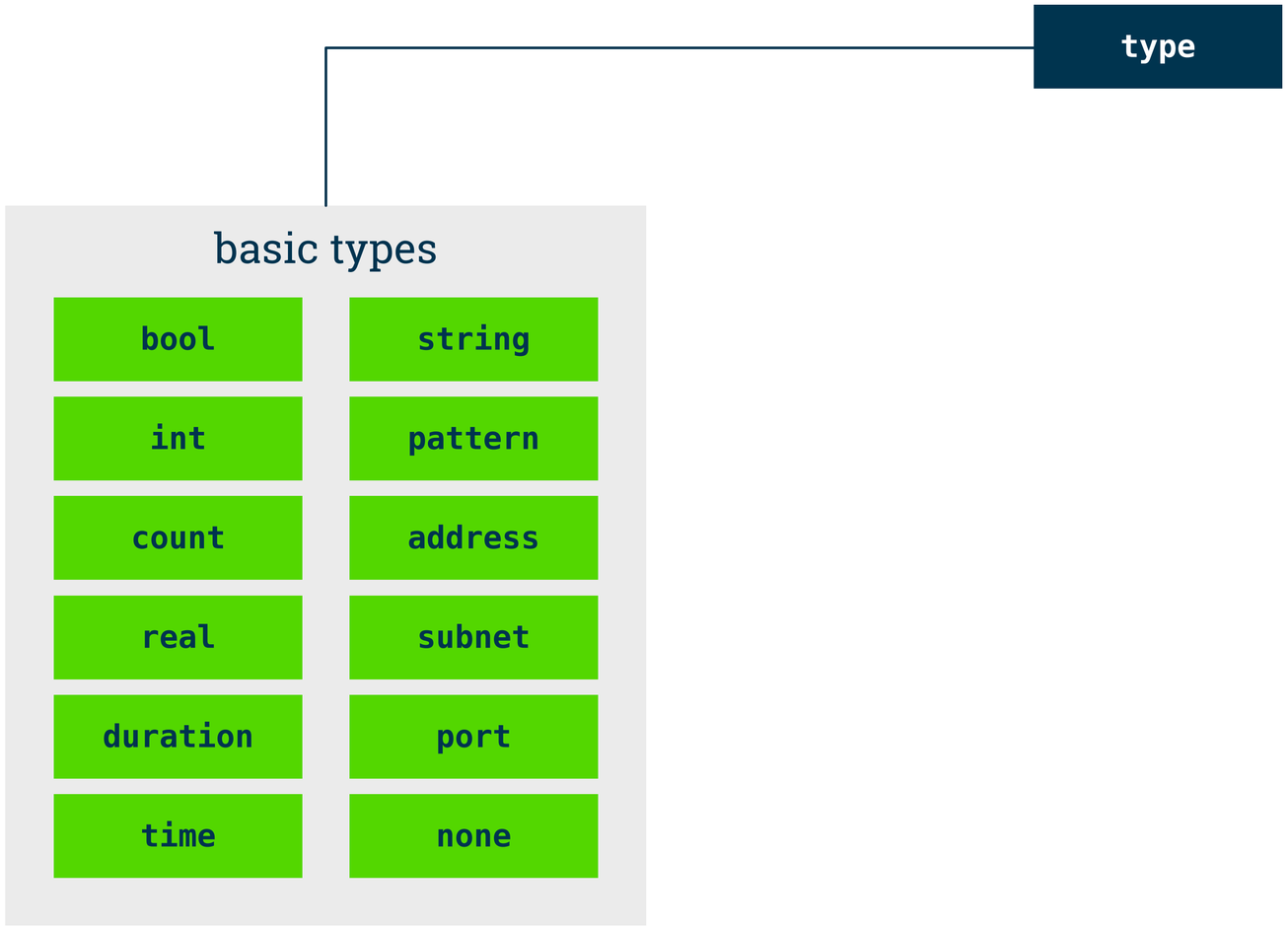
import & export

Data Model

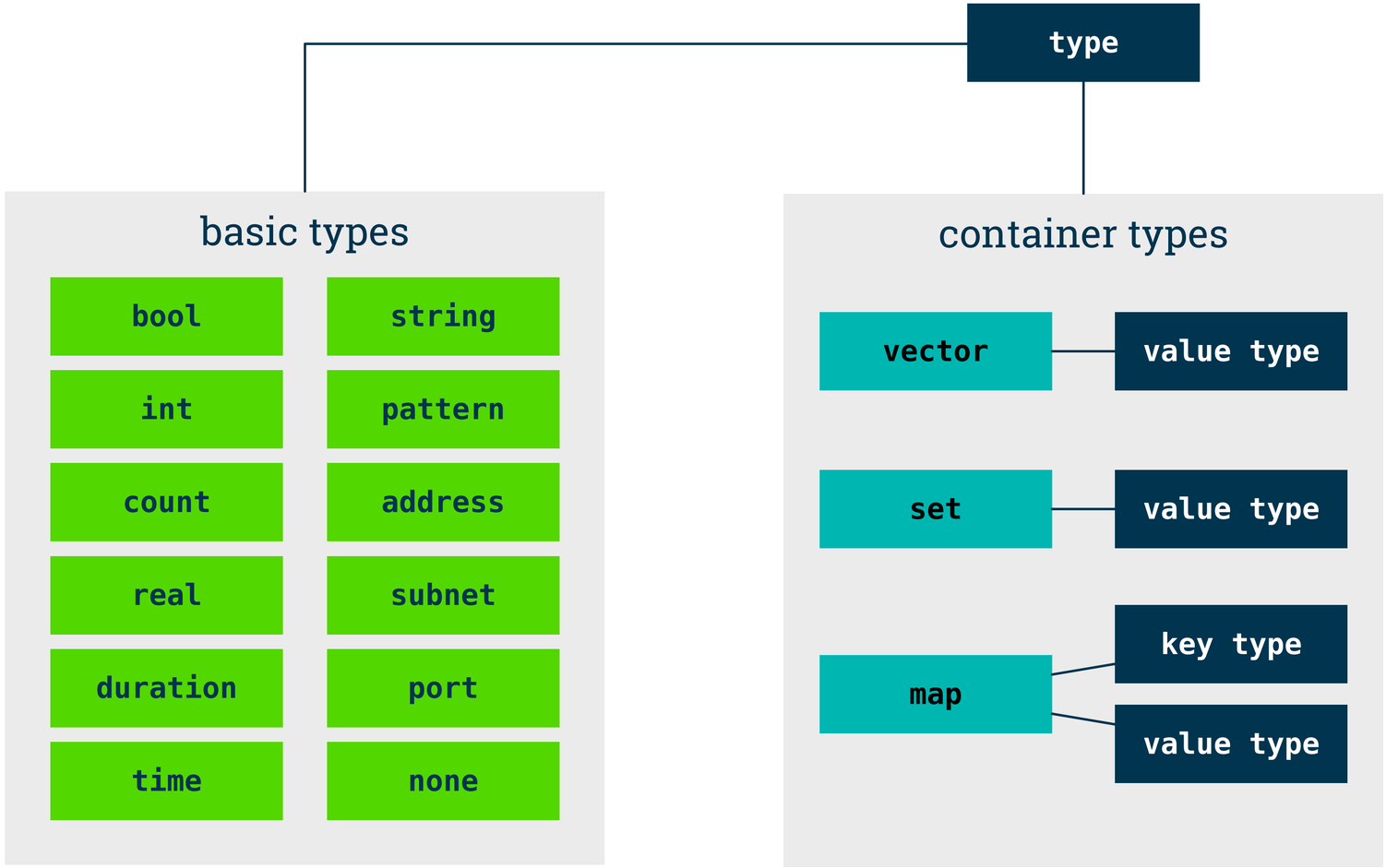
Data Model

`type`

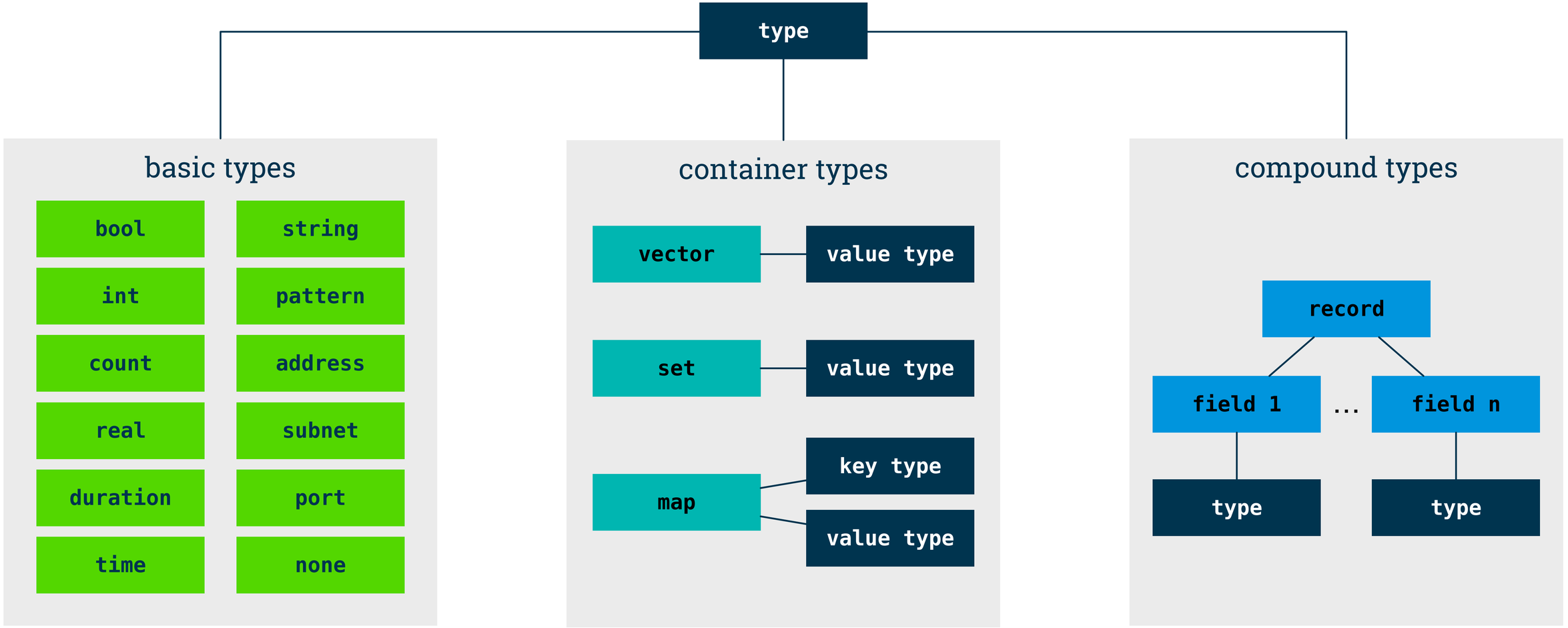
Data Model


type
basic types
bool
string
int
pattern
count
address
real
subnet
duration
port
time
none

Data Model


type
basic types
bool
string
int
pattern
count
address
real
subnet
duration
port
time
none
container types
vector
value type
set
value type
map
key type
value type

Data Model


type
basic types
bool
string
int
pattern
count
address
real
subnet
duration
port
time
none
container types
vector
value type
set
value type
map
key type
value type
compound types
record
field 1
...
field n
type
type

Query Language

Query Language

examples

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )

#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )

orig_bytes >= 1MB && http.method in {“POST”, “PUT”}

#timestamp > 1 hour ago && 6.6.6.6
```

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
```

```
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
```

```
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
```

```
#timestamp > 1 hour ago && 6.6.6.6
```

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )

#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )

orig_bytes >= 1MB && http.method in {“POST”, “PUT”}

#timestamp > 1 hour ago && 6.6.6.6
```

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
#timestamp > 1 hour ago && 6.6.6.6
```

connectives

```
&&  ||  !  (expr)
```

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
#timestamp > 1 hour ago && 6.6.6.6
```

connectives

`&&` `||` `!` `(expr)`

predicates

`LHS` `op` `RHS`

`LHS` `op` `RHS`

`RHS`

`data predicate`

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
#timestamp > 1 hour ago && 6.6.6.6
```

connectives

`&&` `||` `!` `(expr)`

extractors

`x.y.z`	key
`#attr`	attribute
`:T`	type

predicates

`LHS` `op` `RHS`

`LHS` `op` `RHS`

`RHS`

data predicate

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
#timestamp > 1 hour ago && 6.6.6.6
```

connectives

`&&` `||` `!` `(expr)`

predicates

`LHS` `op` `RHS`

`LHS` `op` `RHS`

`RHS`

data predicate

extractors

`x.y.z`	key
`#attr`	attribute
`:T`	type

operators

`==`	`!=`	`~`	`!~`
`<`	`>`	`<=`	`>=`
`in`	`!in`	`ni`	`!ni`

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
#timestamp > 1 hour ago && 6.6.6.6
```

connectives

`&&` `||` `!` `(expr)`

predicates

`LHS` `op` `RHS`

`LHS` `op` `RHS`

`RHS`

data predicate

extractors

`x.y.z`	key
`#attr`	attribute
`:T`	type

operators

`==` `!=` `~` `!~`

`<` `>` `<=` `>=`

`in` `!in` `ni` `!ni`

values

`T, F` `“foo”`

`-42` `/f.*o/`

`7` `6.6.6.6`

`4.2` `10.0.0.0/8`

`42ms` `53/udp`

`2018-07-20` `nil`

Query Language

examples

```
id.orig_h in 192.168.0.0/23 && ( 53/udp || :port > 1024/? )
#type ~ /suricata.*/ && ( src_net !in 10.0.0.0/8 || “evil” )
orig_bytes >= 1MB && http.method in {“POST”, “PUT”}
#timestamp > 1 hour ago && 6.6.6.6
```

SI literals

k	Ki
M	Mi
G	Gi
P	Pi
E	Ei

connectives

`&&` `||` `!` `(expr)`

predicates

`LHS` `op` `RHS`

`LHS` `op` `RHS`

`RHS`

data predicate

extractors

`x.y.z`	key
`#attr`	attribute
`:T`	type

operators

`==`	`!=`	`~`	`!~`
`<`	`>`	`<=`	`>=`
`in`	`!in`	`ni`	`!ni`

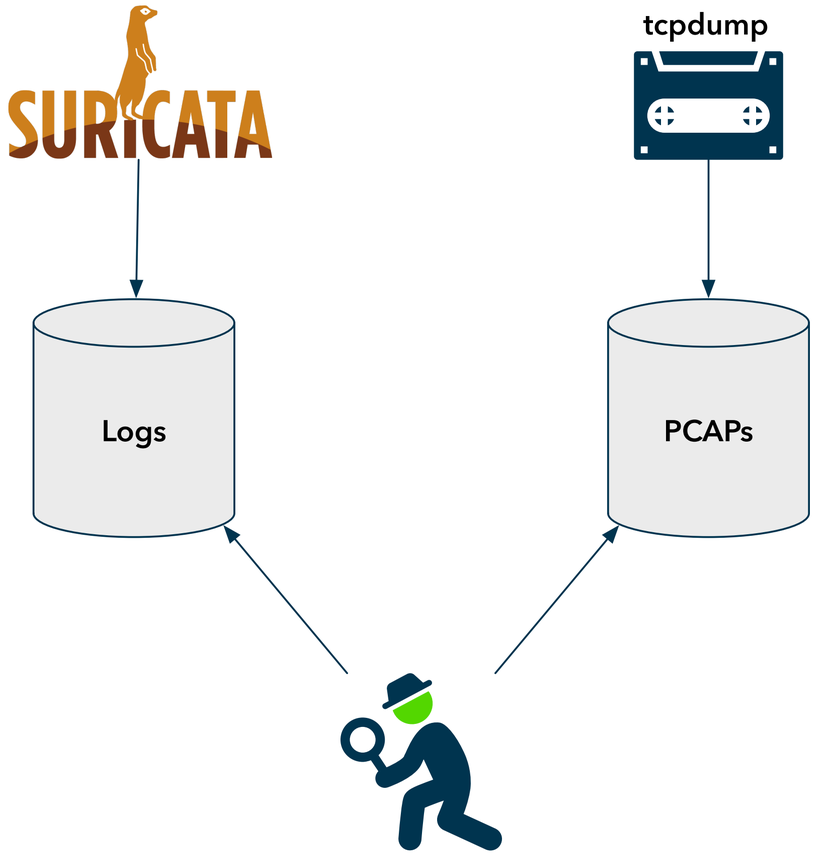
values

`T, F`	`“foo”`
`-42`	`/f.*o/`
`7`	`6.6.6.6`
`4.2`	`10.0.0.0/8`
`42ms`	`53/udp`
`2018-07-20`	`nil`

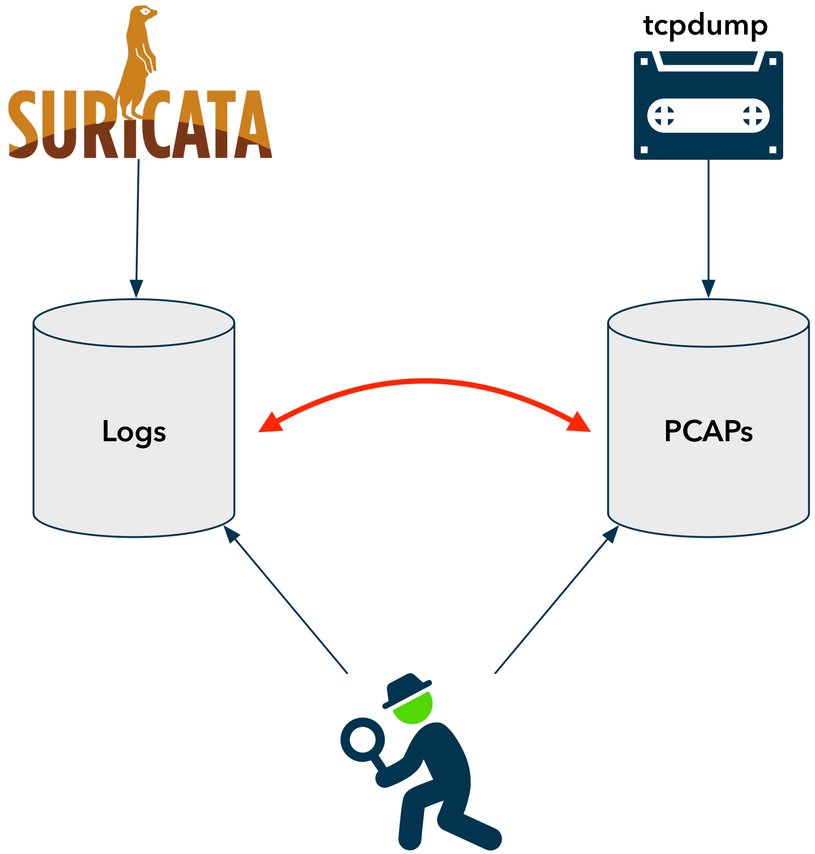
Pivoting

for hunting and forensics

Inter-Tool Pivoting

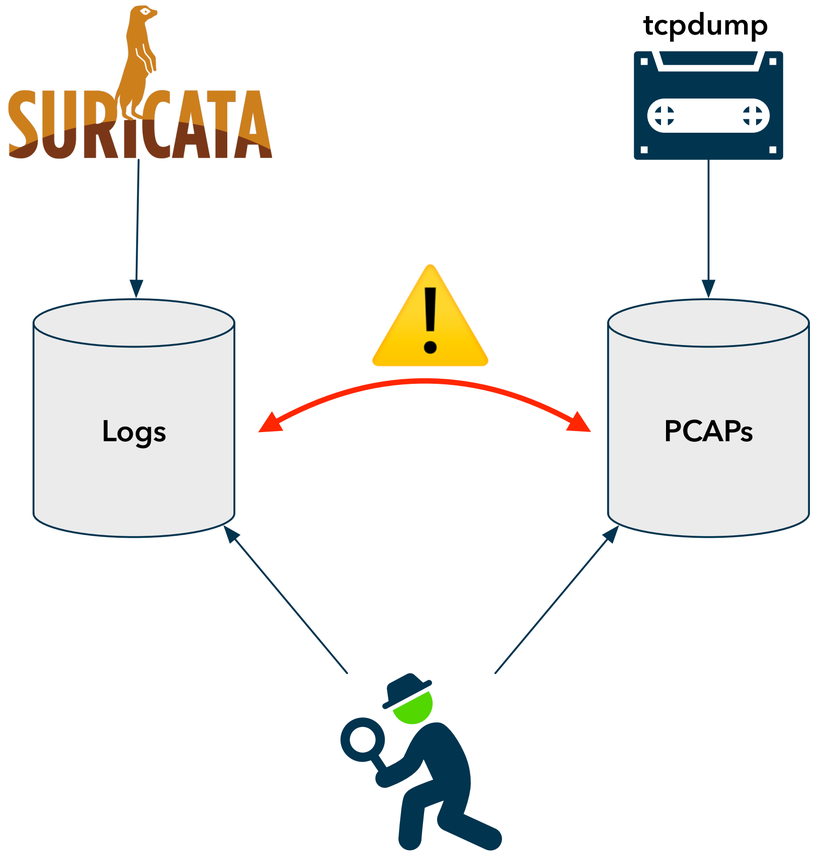


Inter-Tool Pivoting

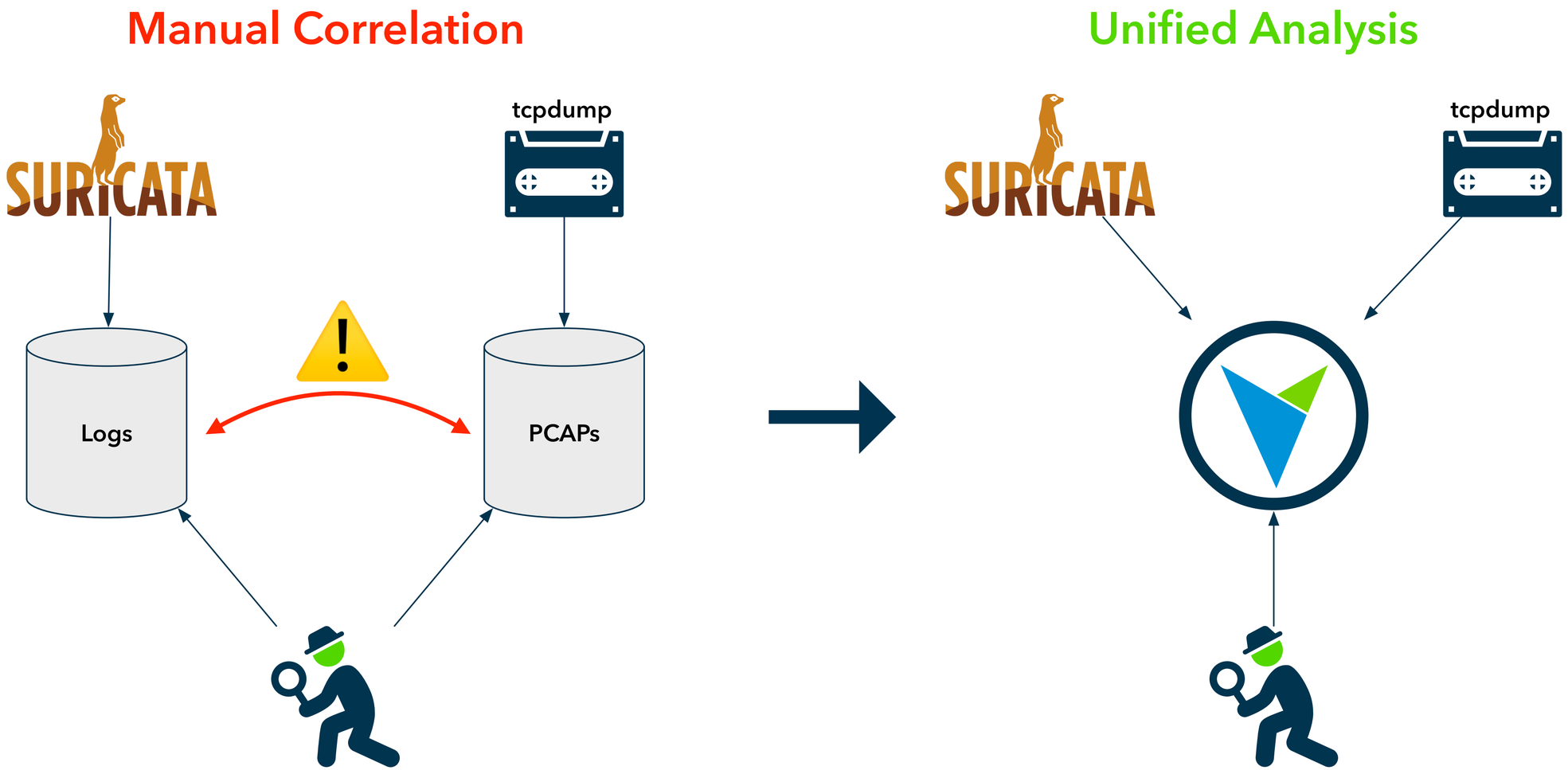


Inter-Tool Pivoting

Manual Correlation

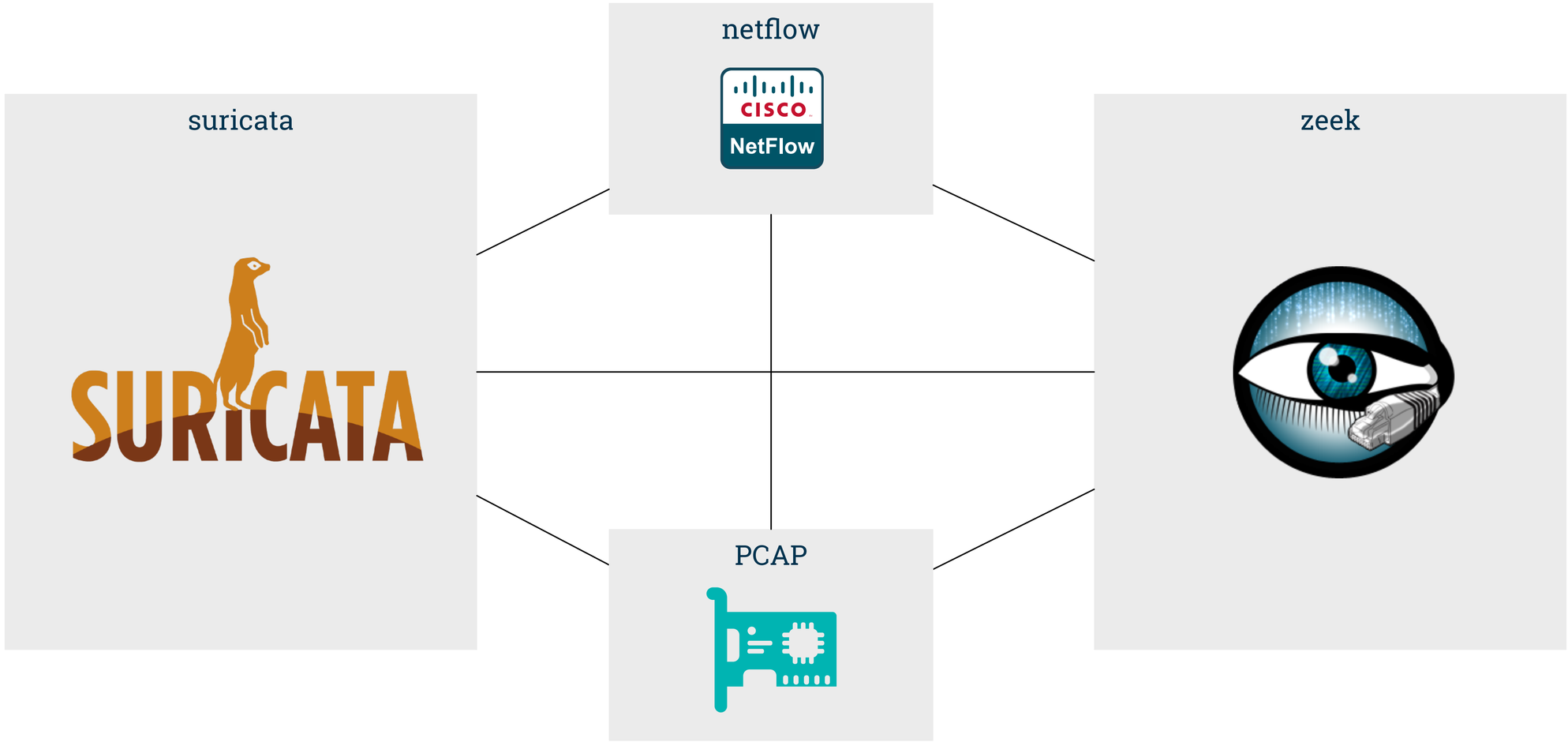


Inter-Tool Pivoting

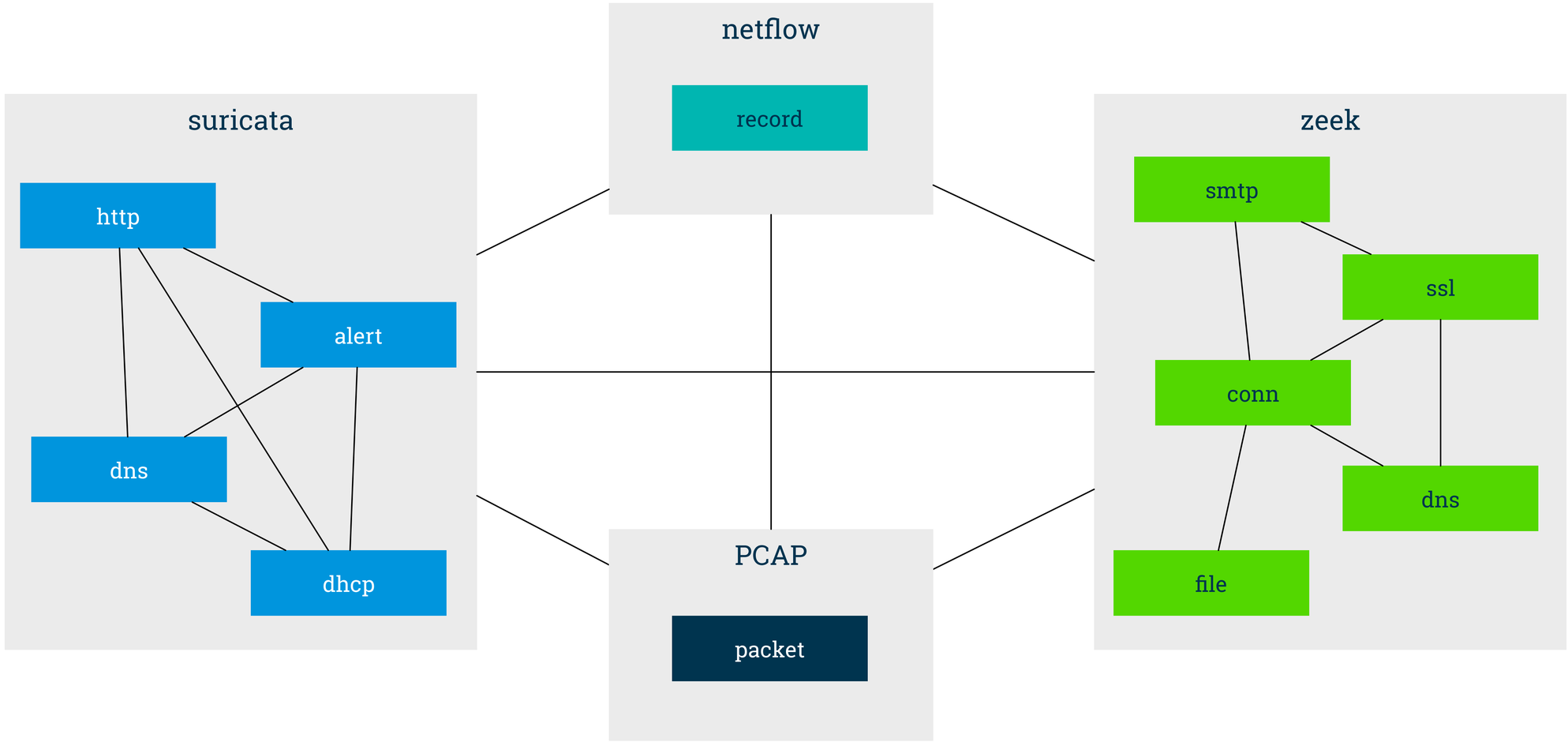


Schema Pivoting

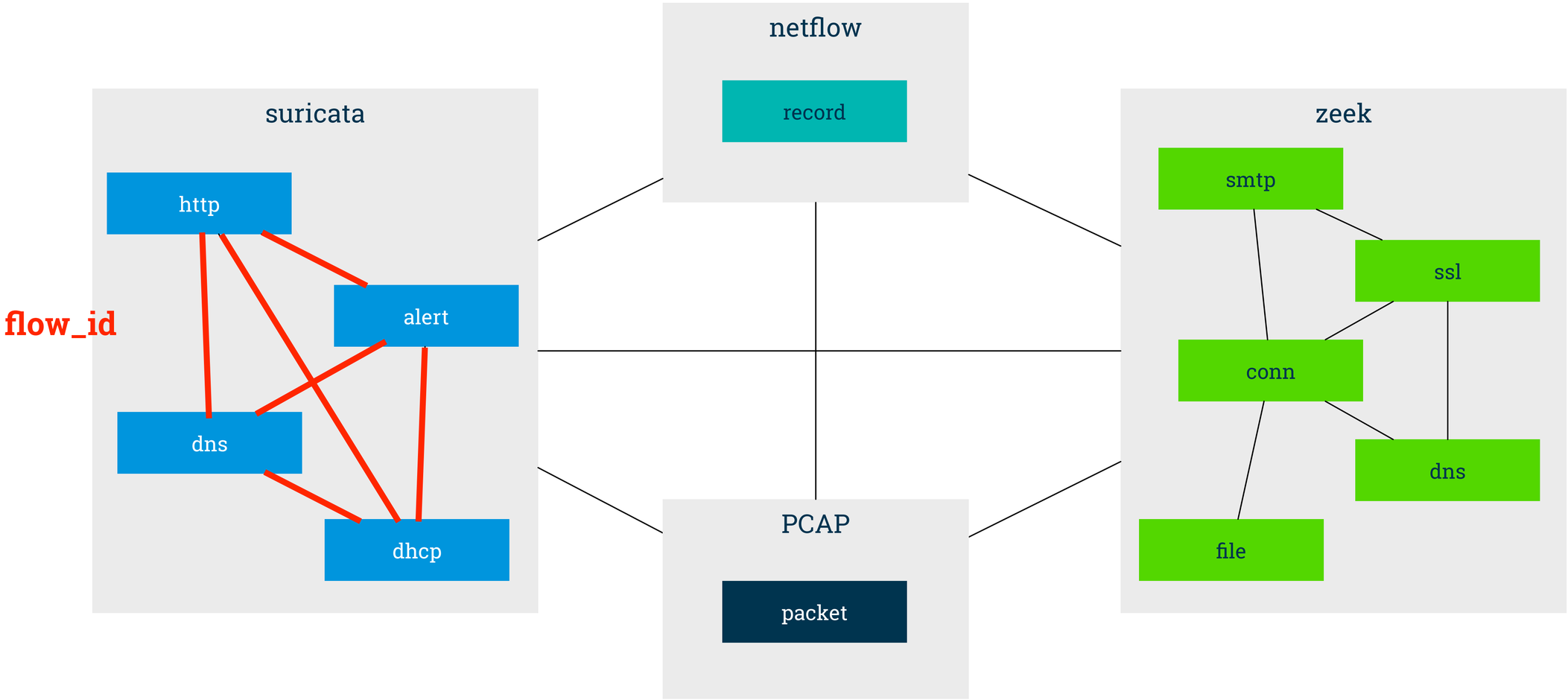
Schema Pivoting



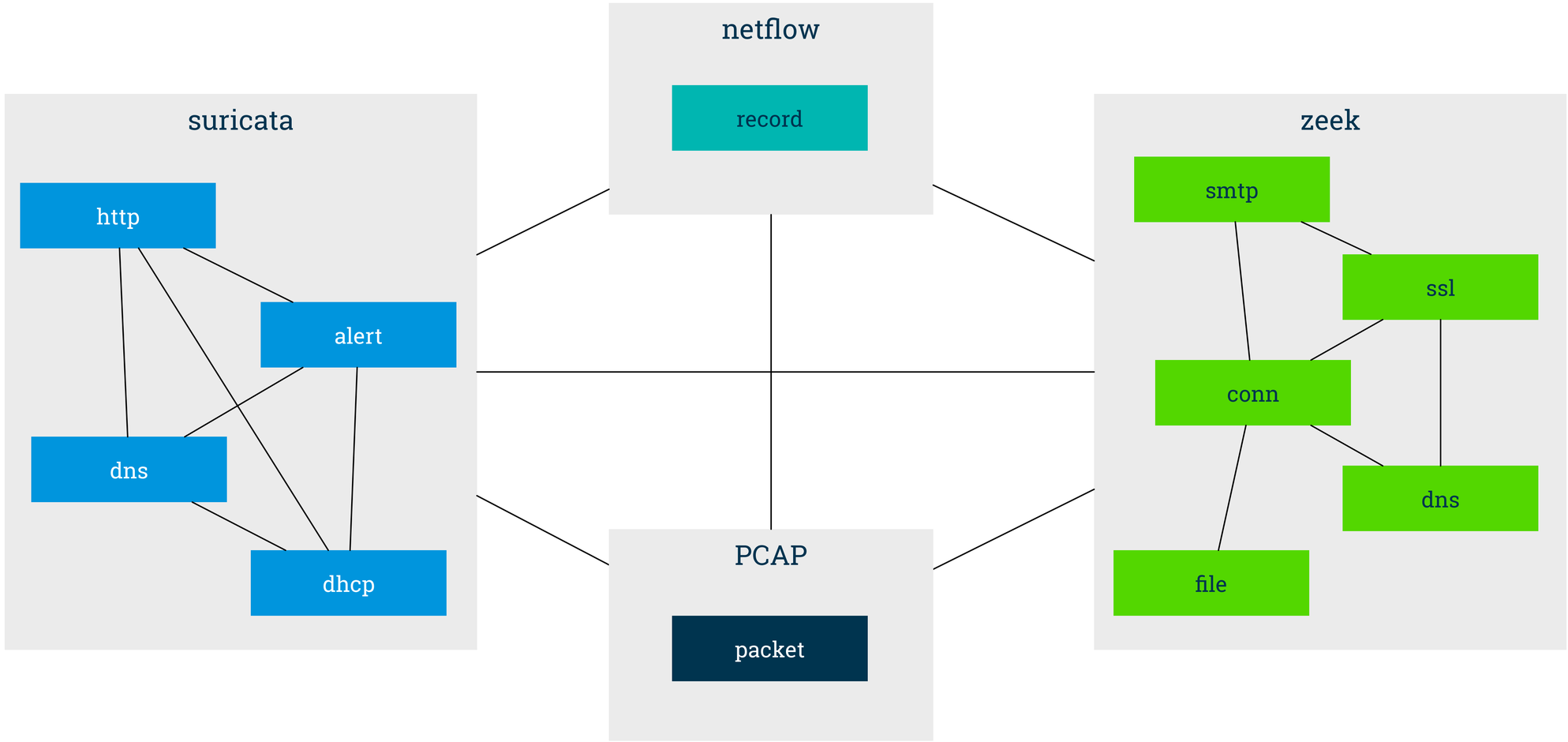
Schema Pivoting



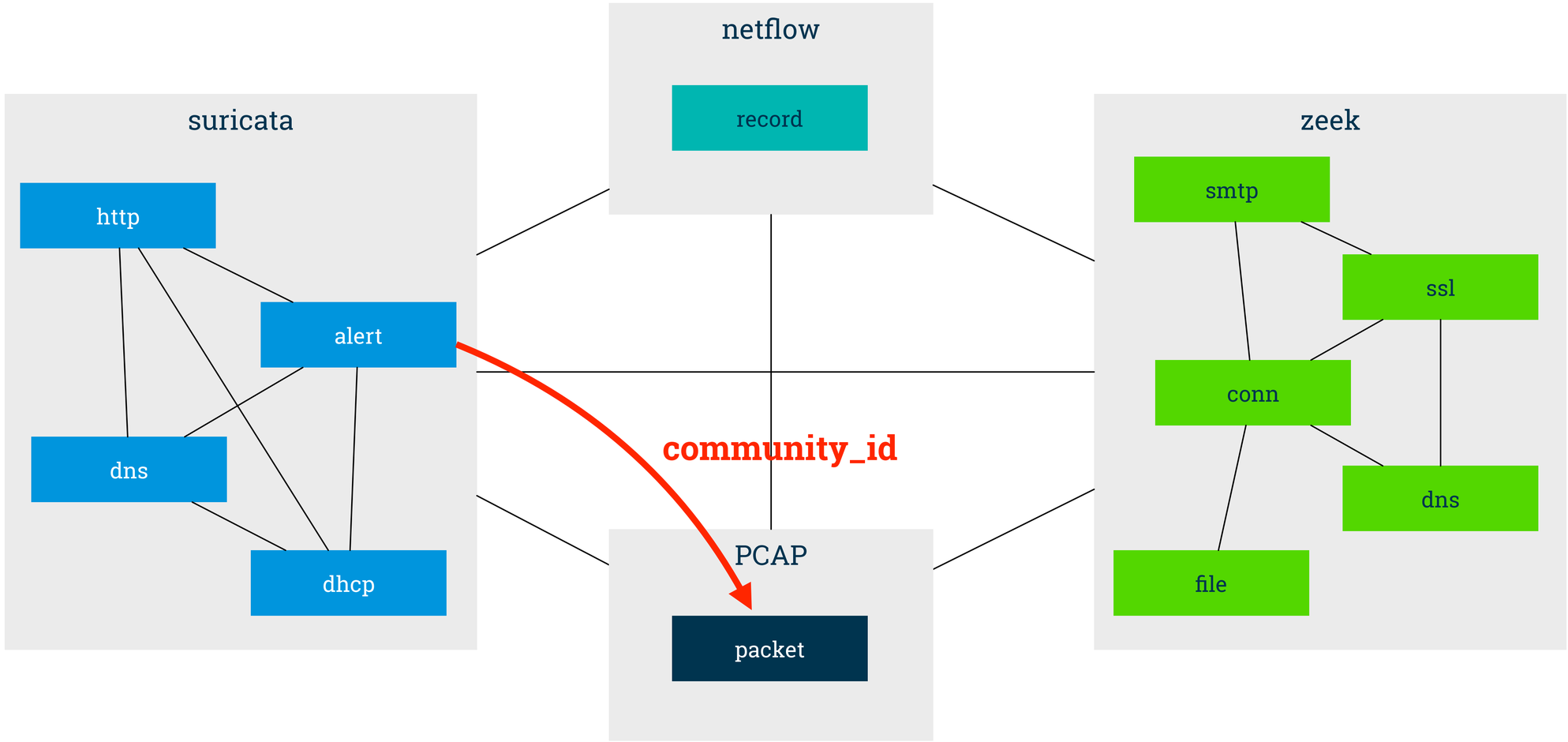
Schema Pivoting



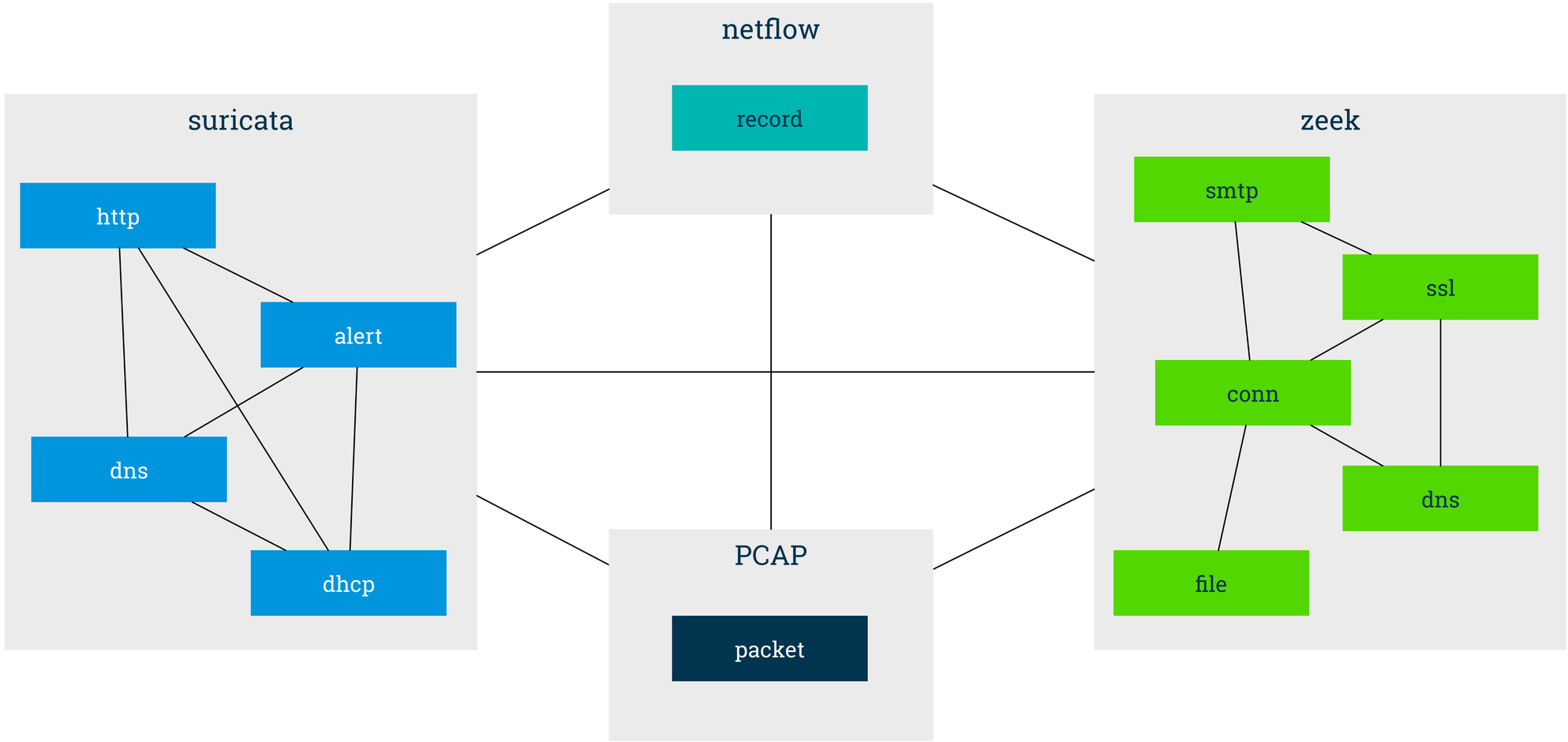
Schema Pivoting



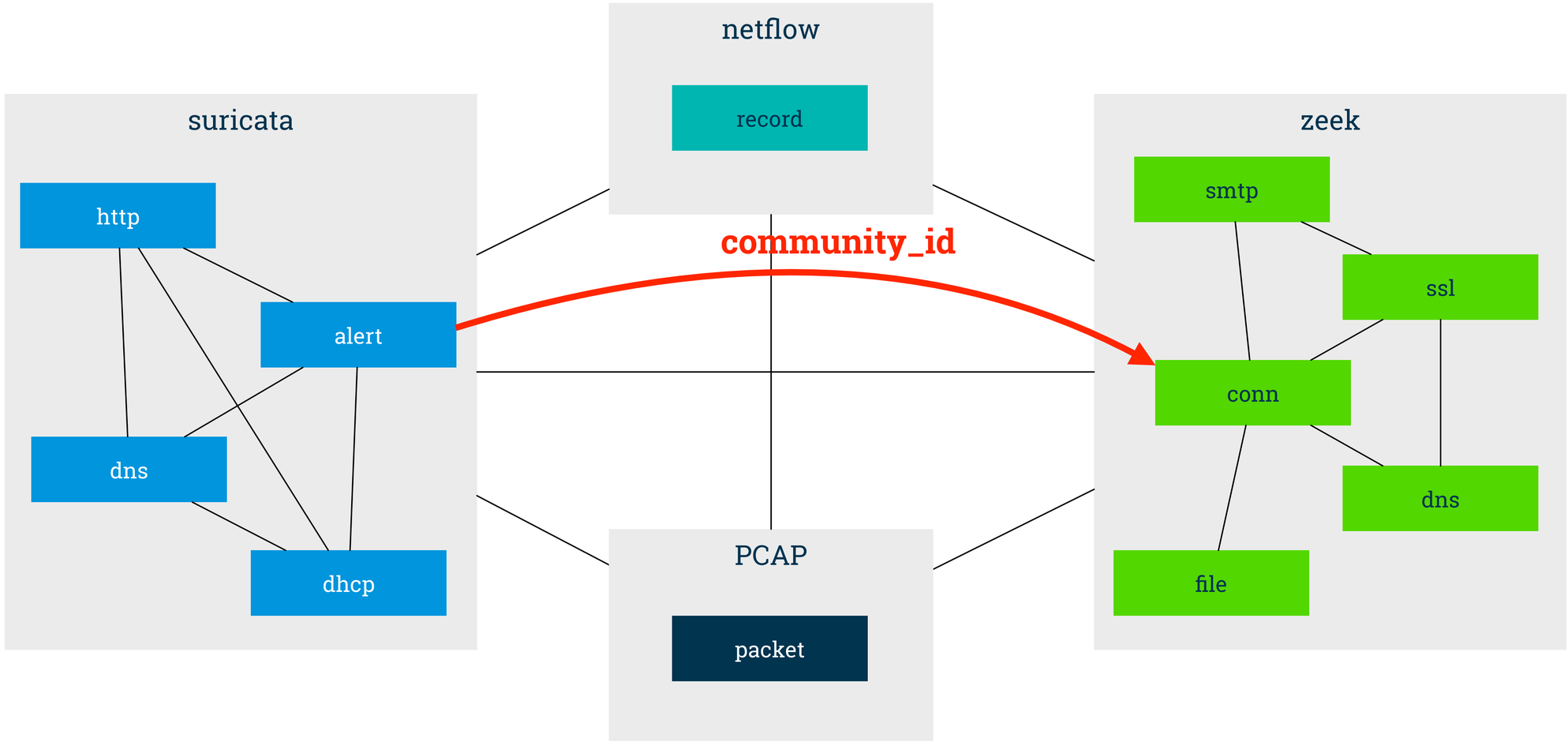
Schema Pivoting



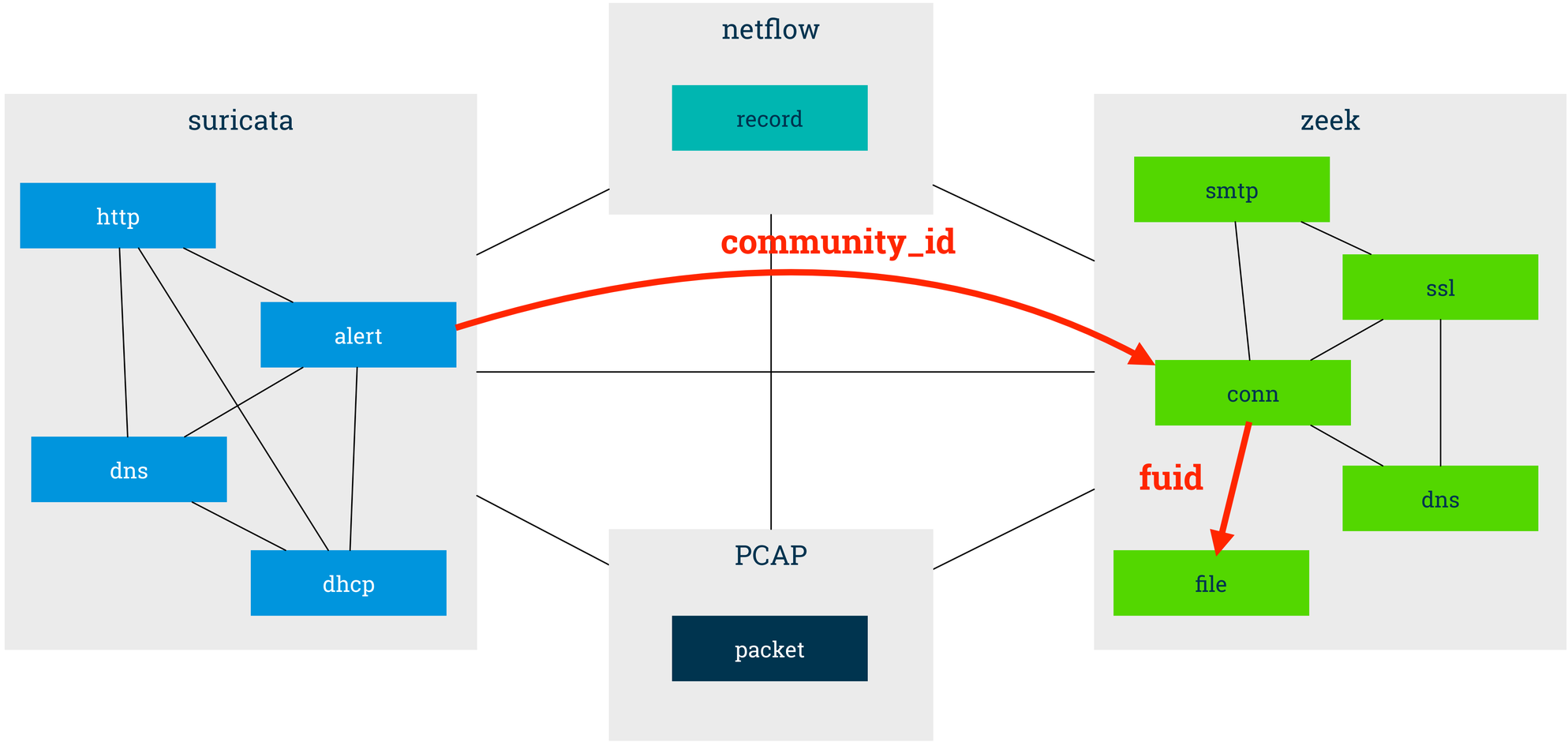
Schema Pivoting



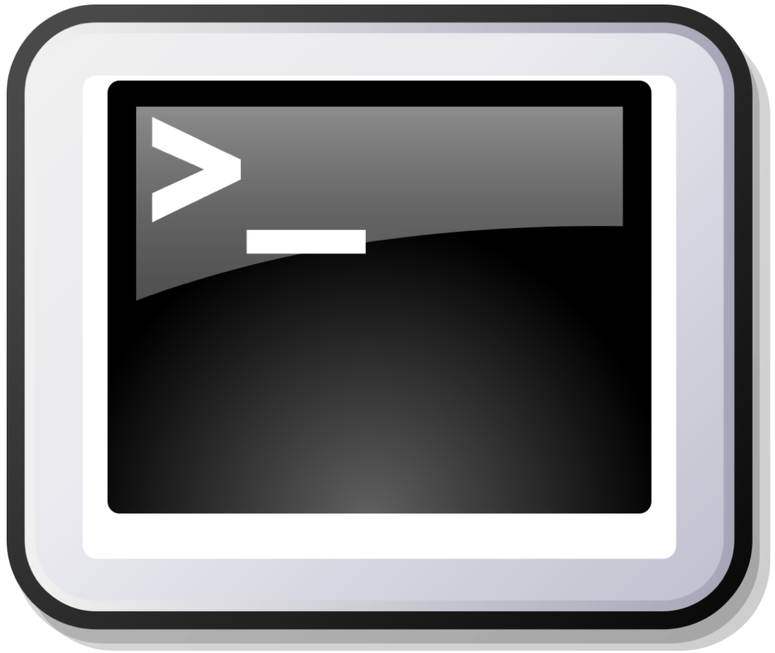
Schema Pivoting



Schema Pivoting



Demo



pivoting

Benchmark

VAST vs. ElasticSearch

Testbed

- Apple iMac 2017
- **CPU**: 4,2 GHz Core i7
  - 4 cores, HT enabled
- **RAM**: 16 GB DDR4
- **SSD**: 250 GB

- **Version**: git master
- **Compiler**: Clang 9
- Build type: release

- **Version**: 7.4.1
- Installed via Homebrew
- Filebeat import, no Logstash
- ML module disabled

Benchmarking Dataset

PCAP

- Trace: M57 2009/11/18
- Enterprise traffic
- ~872k packets

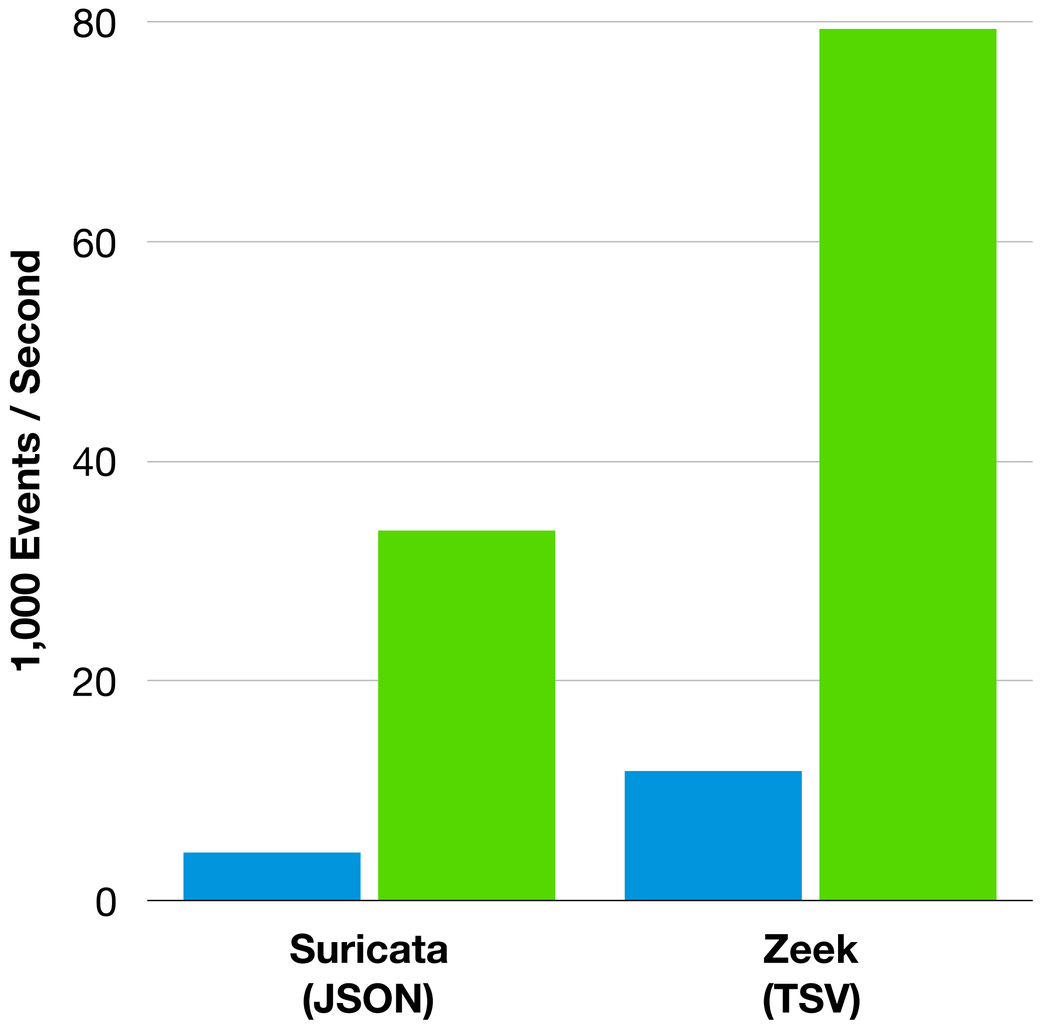
Suricata

- Format: EVE JSON
- ~37k events
- Concatenated 42 times

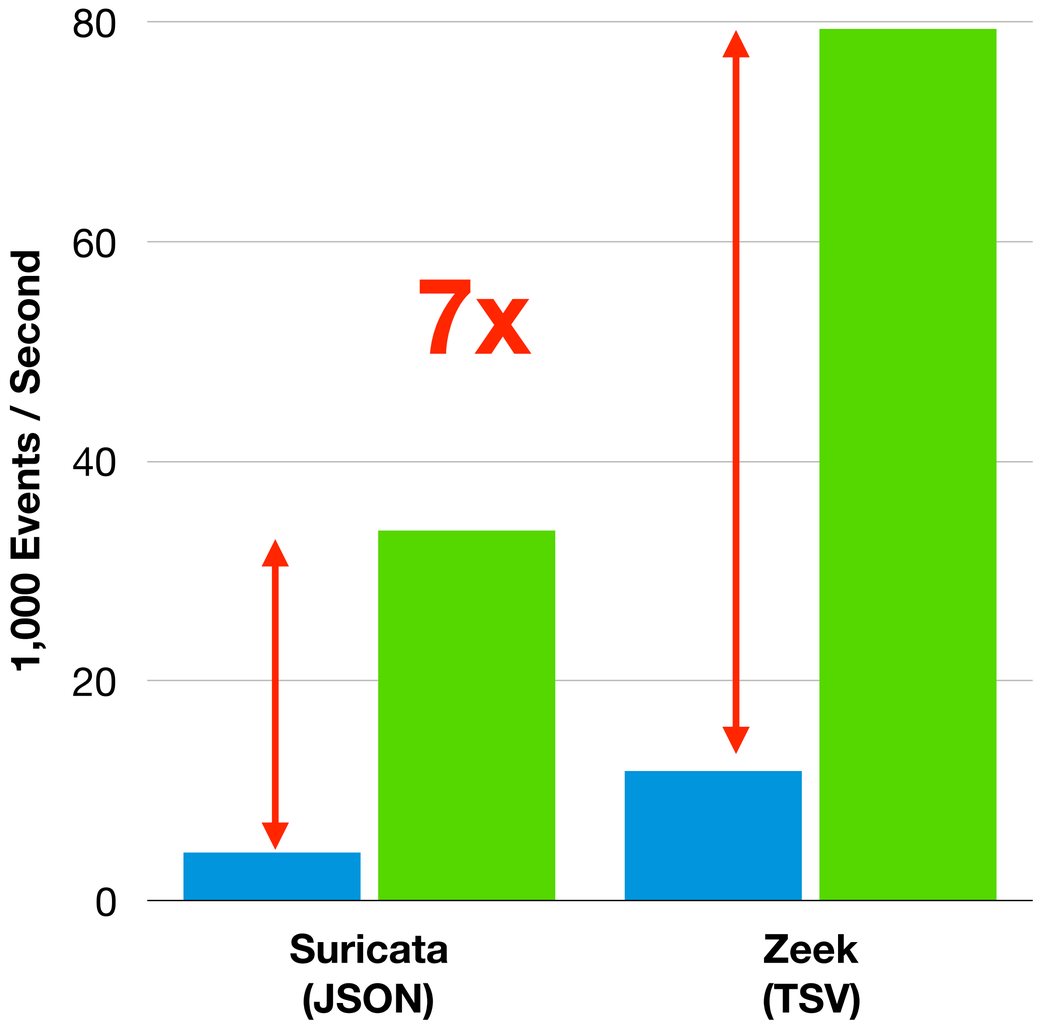
Zeek

- Format: TSV
- Event subset: conn, file, http, dns, ssl
- ~28k events
- Concatenated 180 times

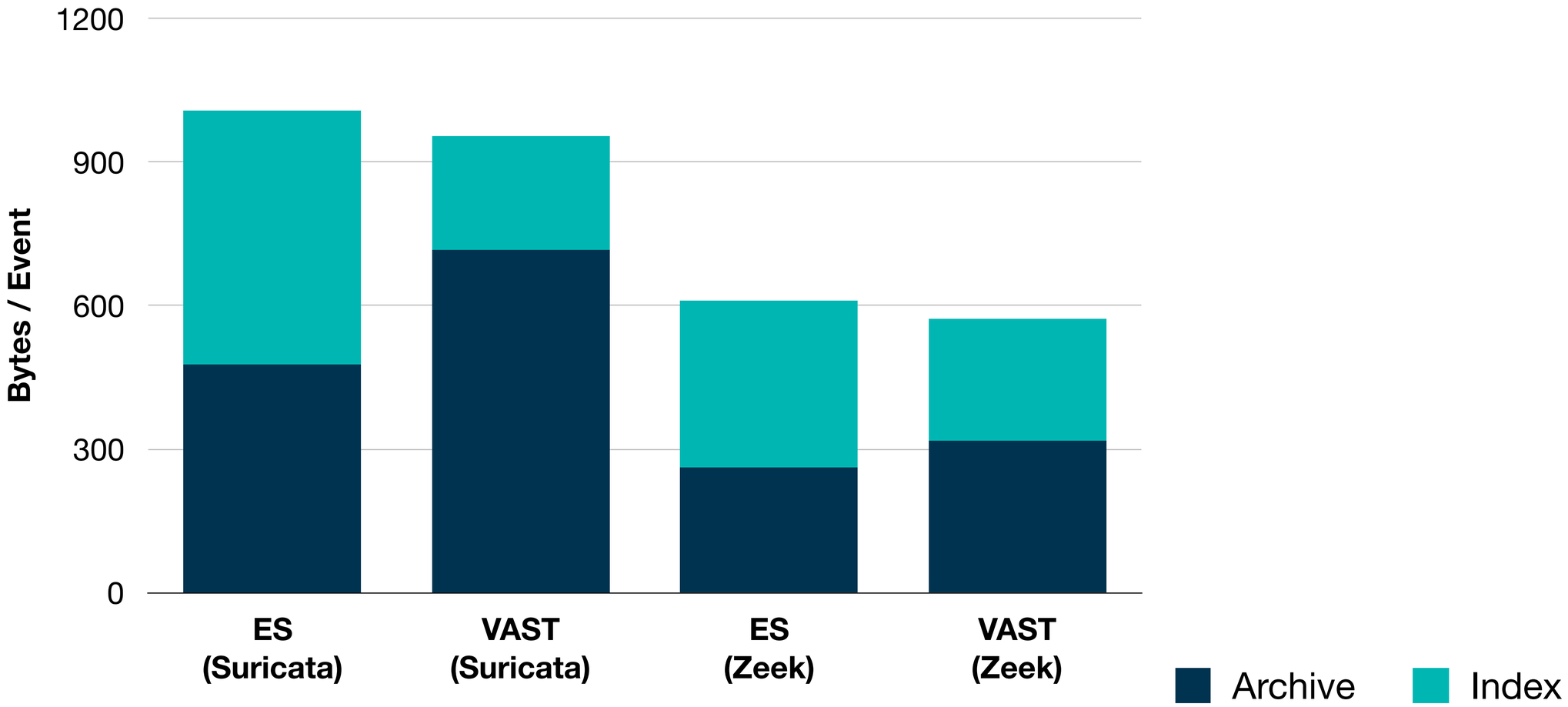
Ingestion Throughput



Ingestion Throughput

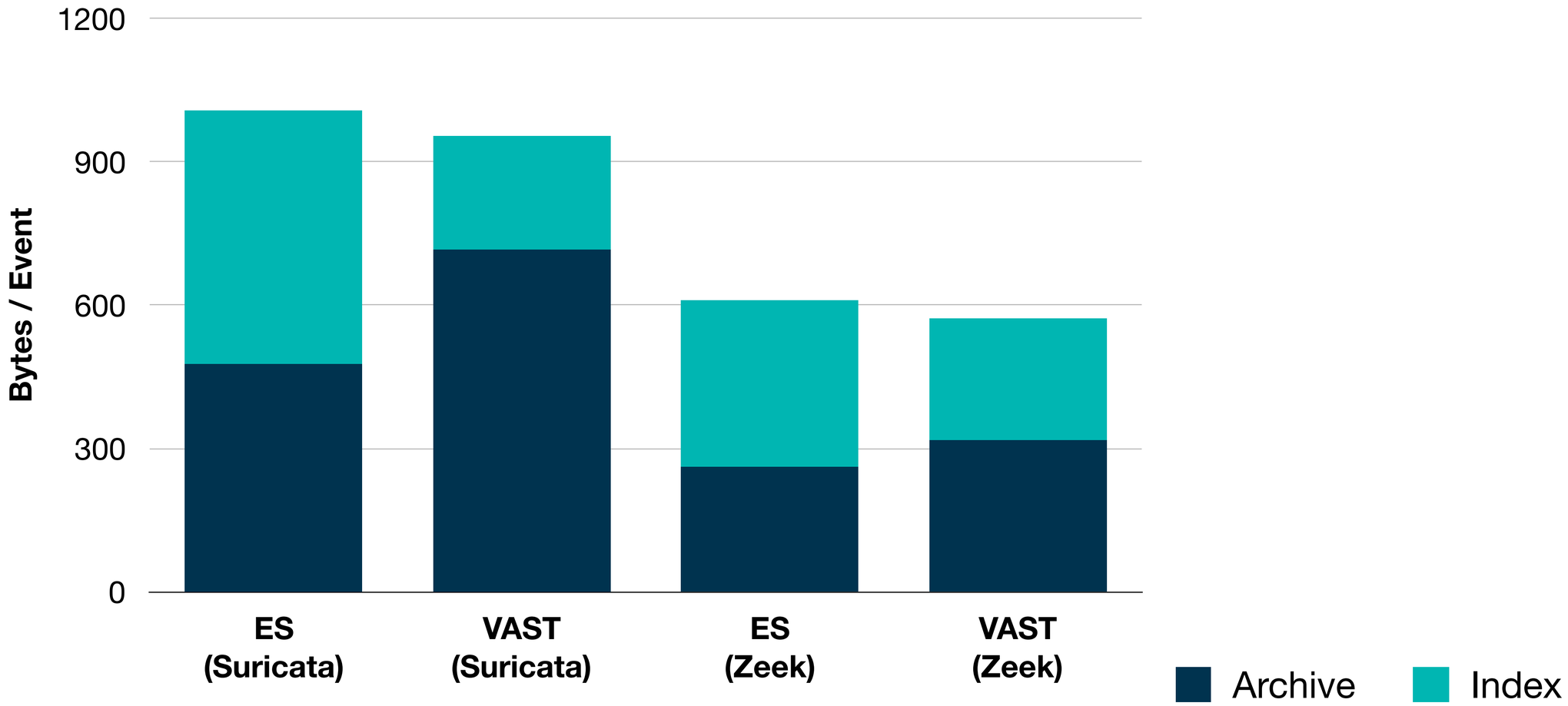


Storage Utilization



Storage Utilization

VAST not yet optimized *at all*



Summary

- **VAST**: accelerated **threat-hunting** and **forensics**
- **Rich data model**, flexible **query language**
- First-class SURICATA support
- **Schema Pivoting**: quickly locate context for related data
- Implementation in **C++20** and the **actor model**
- Comes with a **BSD license**

VAST

github.com/tenzir/vast

TENZIR

info@tenzir.com