Karlsruhe Institute of Technology

The Intelligence Framework Update

Jan Grashöfer

DECENTRALIZED SYSTEMS AND NETWORK SERVICES RESEARCH GROUP (DSN)
INSTITUTE OF TELEMATICS, FACULTY OF INFORMATICS

```
[ 65%] Building CXX object src/analyzer/protocol/teredo/CMakeFiles/plugin-Bro-Teredo.di
[ 65%] Building CXX object src/analyzer/protocol/teredo/CMakeFiles/plugin-Bro-Teredo.di
[ 65%] Building CXX object src/analyzer/protocol/teredo/CMakeFiles/plugin-Bro-Teredo.di
[ 65%] Building CXX object src/analyzer/protocol/teredo/CMakeFiles/plugin-Bro-Teredo.di
[ 65%] Linking CXX static library libplugin-Bro-Teredo.a
make[3]: Leaving directory '/home/jgras/devel/bro/build'
[ 65%] Built target plugin-Bro-Teredo
make[3]: Entering directory '/home/jgras/devel/bro/build'
Scanning dependencies of target plugin-Bro-UDP
make[3]: Leaving directory '/home/jgras/devel/bro/build'
make[3]: Entering directory '/home/jgras/devel/bro/build'
[ 66%] Building CXX object src/analyzer/protocol/udp/CMakeFiles/plugin-Bro-UDP.dir/UDP.
[ 66%] Building CXX object src/analyzer/protocol/udp/CMakeFiles/plugin-Bro-UDP.dir/Plug
[ 66%] Building CXX object src/analyzer/protocol/udp/CMakeFiles/plugin-Bro-UDP.dir/ever
[ 66%] Building CXX object src/analyzer/protocol/udp/CMakeFiles/plugin-Bro-UDP.dir/ever
[ 66%] Linking CXX static library libplugin-Bro-UDP.a
make[3]: Leaving directory '/home/jgras/devel/bro/build'
[ 66%] Built target plugin-Bro-UDP
make[3]: Entering directory '/home/jgras/devel/bro/build'
Scanning dependencies of target plugin-Bro-XMPP
make[3]: Leaving directory '/home/jgras/devel/bro/build'
```

```
37  hook extend_match(info: Info, s: Seen, items: set[Item])
38      {
39      local matches = |items|;
40      for ( item in items )
41          {
42          local meta = item$meta;
43          if ( meta$expire > 0 sec &&
44               meta$last_match + meta$expire < network_time() &&
45               ! hook single_item_expired(item) )
46              {
47              # Item already expired
48              --matches;
49              remove(item, F);
50              next;
51              }
52
53          # Update last match
54          item$meta$last_match = network_time();
55          insert(item);
56          }
57      if ( matches < 1 )
58          break;
```

About me

- Studied Computer Science in Germany:
  - 2013: B.Sc. @ TU Dortmund
  - 2016: M.Sc. @ KIT
- Since 2017: Researcher & PhD Candidate
  - DSN Research Group @ KIT
- First contact: 2015
  - Technical Student @ CERN
- My contributions (so far):
  - Intelligence framework update
  - AF_Packet plugin
  - Miscellaneous improvements
  - Some custom scripts

jan.grashoefer@kit.edu
github.com/J-Gras
Linkedin.com/in/jan-grashoefer

Common Ground

example.intel:

```
#field indicator indicator_type meta.source meta.desc meta.url
bro.org Intel::DOMAIN test-src1 domain for testing http://src1.example.com/id-x
bro.org Intel::DOMAIN test-src2 domain for testing http://src2.example.com/id-y
```

local.bro:

```
@load frameworks/intel/seen
@load frameworks/intel/do_notice

const feed_directory = "/usr/local/bro/feeds";

redef Intel::read_files += {
          feed_directorty + "/example.intel",
          #...
};
```

intel.log:

```
#fields ts   uid    [id] seen.indicator seen.indicator_type seen.where seen.node matched sources
#types  time string [id]   string            enum           enum   string  set[enum] set[string]
[ts] CeNmIl1iWXscljuRw4 … bro.org Intel::DOMAIN HTTP::IN_HOST_HEADER bro Intel::DOMAIN test-src1
[ts] CeNmIl1iWXscljuRw4 … bro.org Intel::DOMAIN HTTP::IN_HOST_HEADER bro Intel::DOMAIN test-src1
[ts] CeNmIl1iWXscljuRw4 … bro.org Intel::DOMAIN HTTP::IN_HOST_HEADER bro Intel::DOMAIN test-src1
```

Outline

- Basics
  - Data Model
  - Ingesting Intel
  - Matching Intel
- Intel Framework Update
- Intel-Extensions
- Future Work

Overall Goal:

Shed some light on the ideas behind the Intelligence Framework

→ Allow users effectively apply it

DSN Research Group
Institute of Telematics, Faculty of Informatics

Data Model

Data Model

Indicator Type	Example
ADDR	192.0.2.42, 2001:db8::23
SUBNET	192.0.2.0/24, 2001:db8::/32
URL	~~**http://**~~example.com/test/
SOFTWARE	Mozilla/5.0…
EMAIL	malicious@example.com
DOMAIN	www.example.com
USER_NAME	not used
CERT_HASH	38762cf...bb7f0a
PUBKEY_HASH	ee4aa5…0a750c
FILE_HASH	5bd9d8…39b8d1
FILE_NAME	infected.pdf

Data Model

Indicator Type	Example
ADDR	192.0.2.42, 2001:db8::23
SUBNET	192.0.2.0/24, 2001:db8::/32
URL	~~**http://**~~example.com/test/
SOFTWARE	Mozilla/5.0…
EMAIL	malicious@example.com
DOMAIN	www.example.com
USER_NAME	not used
CERT_HASH	38762cf...bb7f0a
PUBKEY_HASH	ee4aa5…0a750c
FILE_HASH	5bd9d8…39b8d1
FILE_NAME	infected.pdf

indicator

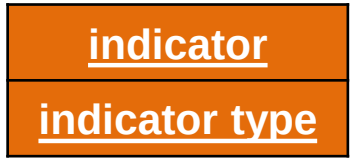
indicator type

File related infos can be disabled:

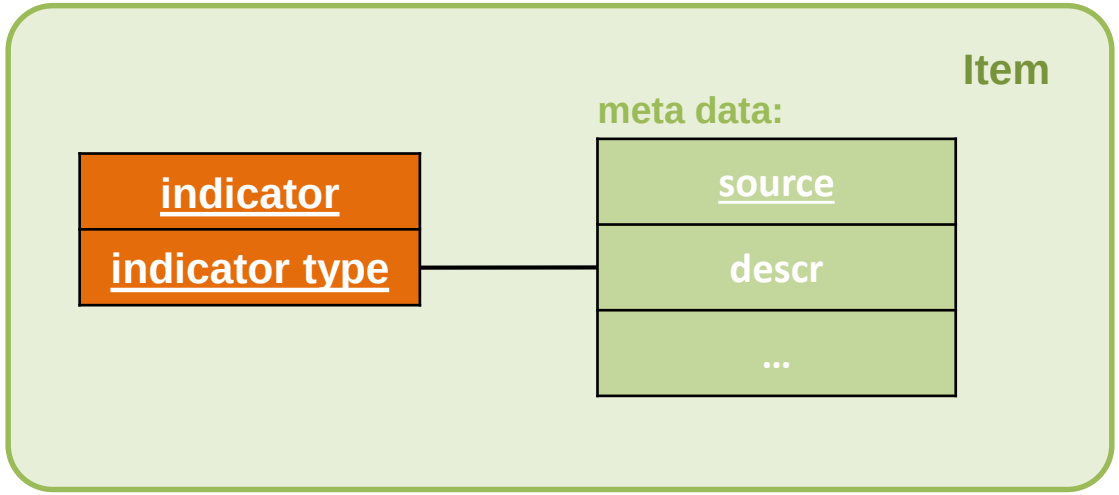
1. start Bro in bare-mode [-b]
2.
```
@unload base/frameworks/intel/files
@load base/init-default
```


DSN Research Group
Institute of Telematics, Faculty of Informatics

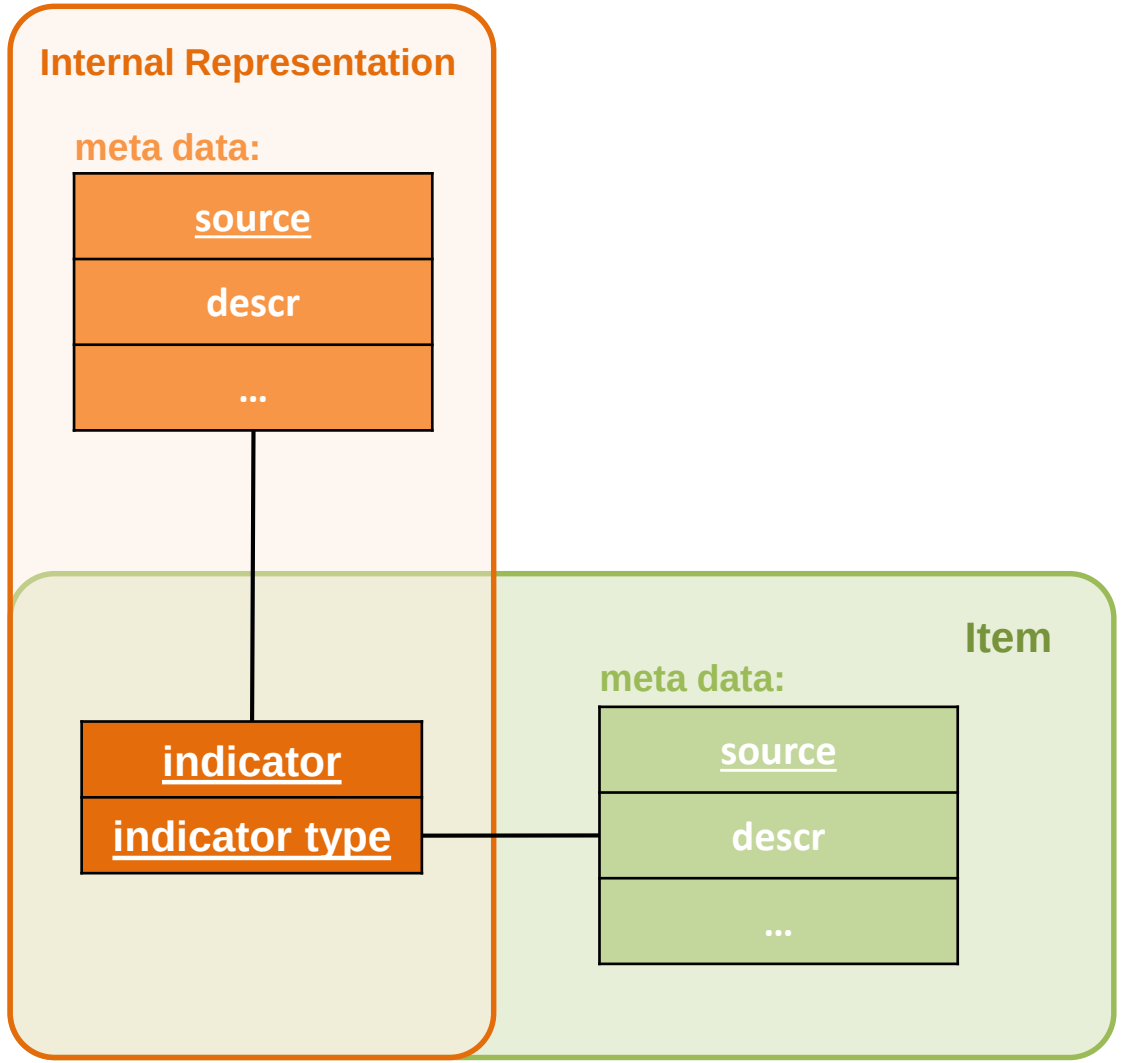
Data Model



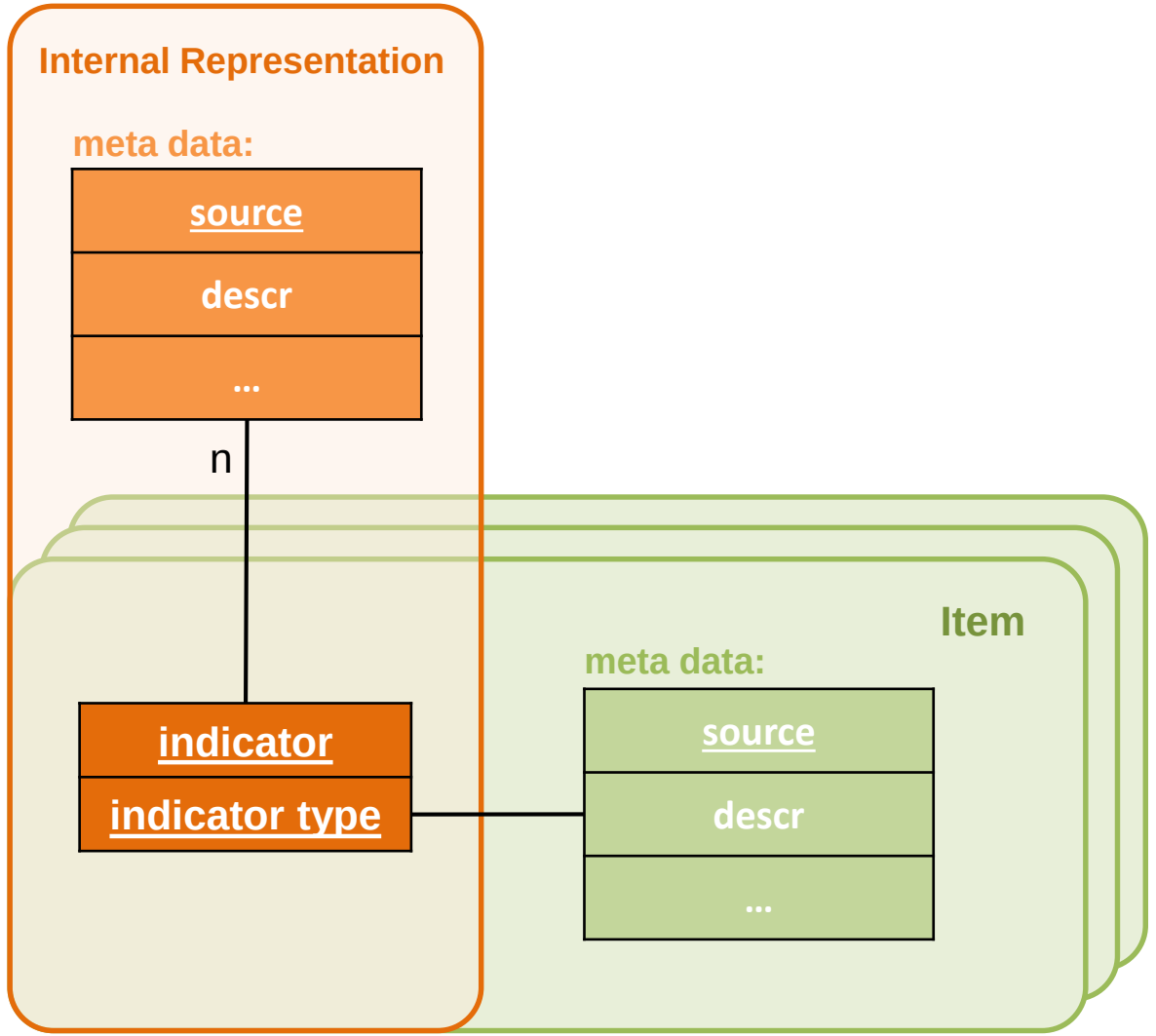
Data Model



Data Model


Internal Representation
meta data:
source
descr
...
indicator
indicator type
Item
meta data:
source
descr
...

Data Model


Internal Representation
meta data:
source
descr
...
n
Item
meta data:
source
descr
...
indicator
indicator type

Ingesting Intel

- Ingested Items compose “in-memory database” of indicators and their corresponding meta data
- Script-Layer perspective: `function Intel::insert(item: Intel::Item)`

- Default approach: Using files (→ Input Framework)

```
event Intel::read_entry([…], item: Intel::Item)
        {
        Intel::insert(item);
        }
```

input.bro:

DSN Research Group
Institute of Telematics, Faculty of Informatics

Ingesting Intel

- Ingested Items compose “in-memory database” of indicators and their corresponding meta data
- Script-Layer perspective: `function Intel::insert(item: Intel::Item)`

- Default approach: Using files (→ Input Framework)

```
event Intel::read_entry([…], item: Intel::Item)
        {
        Intel::insert(item);
        }
```

input.bro:

1. Changes should be atomic (using `mv`)
2. Once an intel file changes, **all** items are reinserted!

Ingesting Intel: Example

example.intel:

```
#field indicator indicator_type meta.source meta.desc meta.url
bro.org Intel::DOMAIN test-src1 domain for testing http://src1.example.com/id-x
```

Internal Representation

indicator:	bro.org
indicator type:	Intel::DOMAIN

meta data:

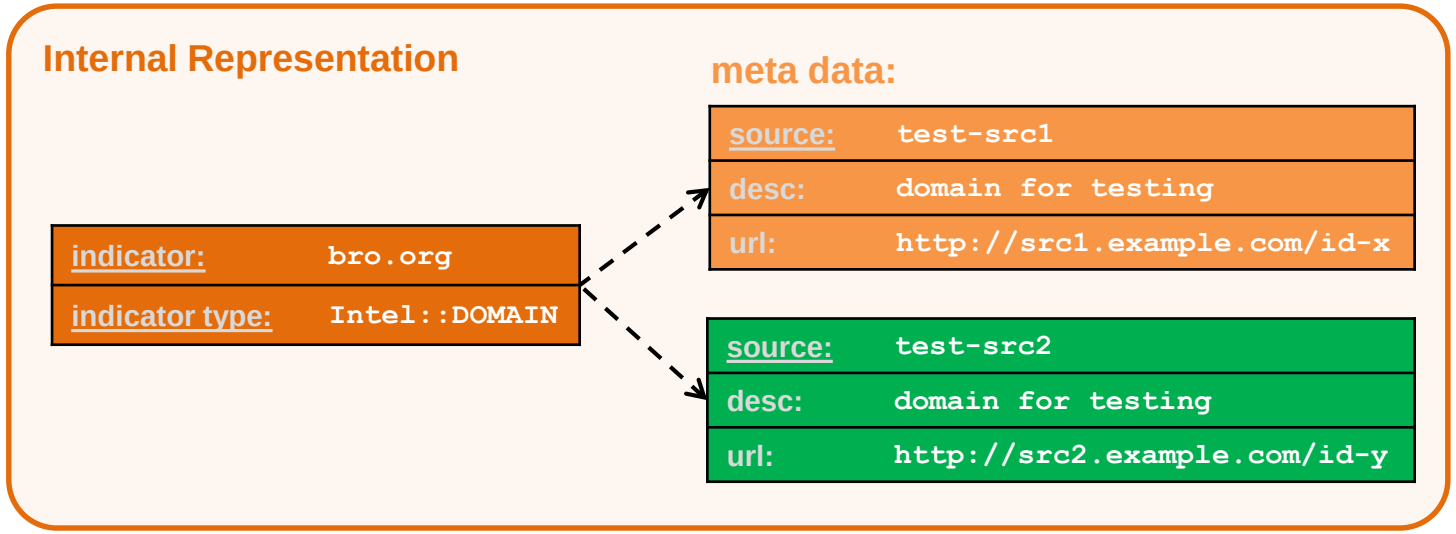
source:	test-src1
desc:	domain for testing
url:	http://src1.example.com/id-x

DSN Research Group
Institute of Telematics, Faculty of Informatics

Ingesting Intel: Example

example.intel:

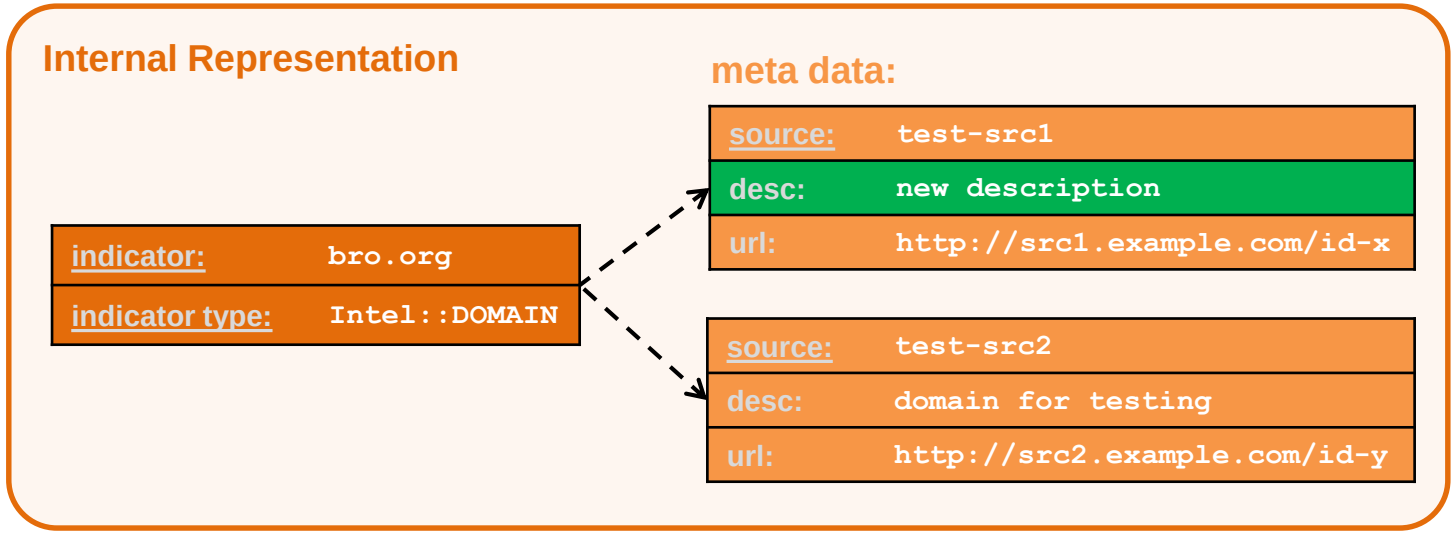
```
#field indicator indicator_type meta.source meta.desc meta.url
bro.org Intel::DOMAIN test-src1 domain for testing http://src1.example.com/id-x
bro.org Intel::DOMAIN test-src2 domain for testing http://src2.example.com/id-y
```



Ingesting Intel: Example

example.intel:

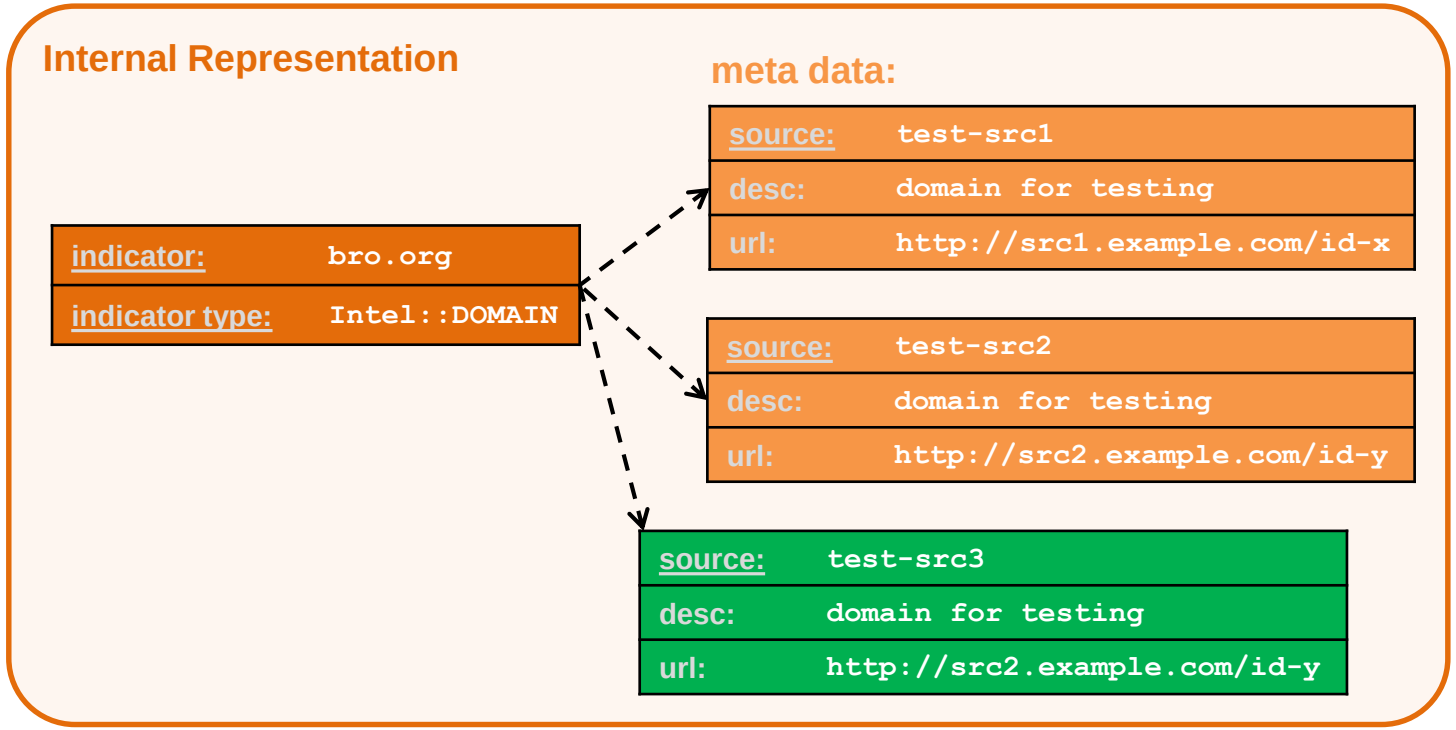
```
#field indicator indicator_type meta.source meta.desc meta.url
bro.org Intel::DOMAIN test-src1 new description    http://src1.example.com/id-x
bro.org Intel::DOMAIN test-src2 domain for testing http://src2.example.com/id-y
```



Ingesting Intel: Example

example.intel:

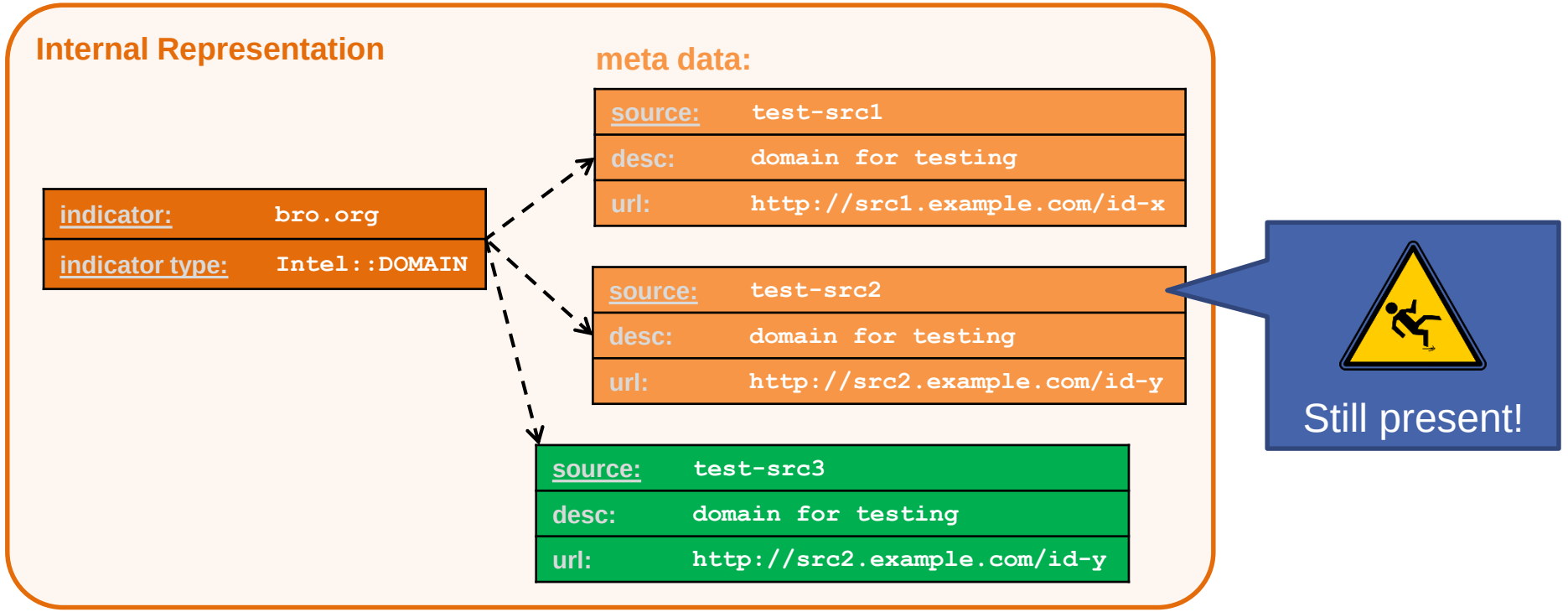
```
#field indicator indicator_type meta.source meta.desc meta.url
bro.org Intel::DOMAIN test-src1 new description    http://src1.example.com/id-x
bro.org Intel::DOMAIN test-src3 domain for testing http://src2.example.com/id-y
```



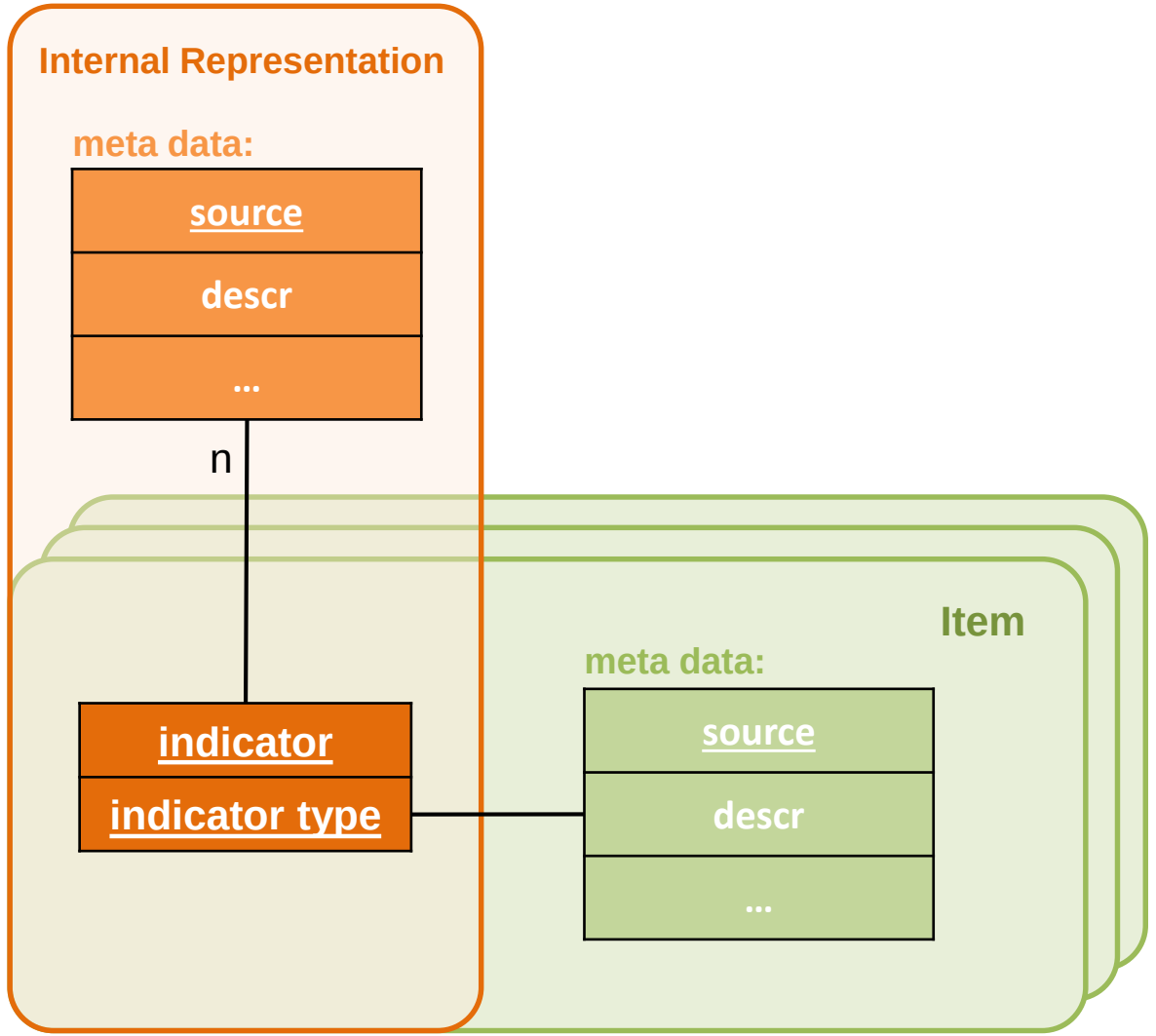
Ingesting Intel: Example

example.intel:

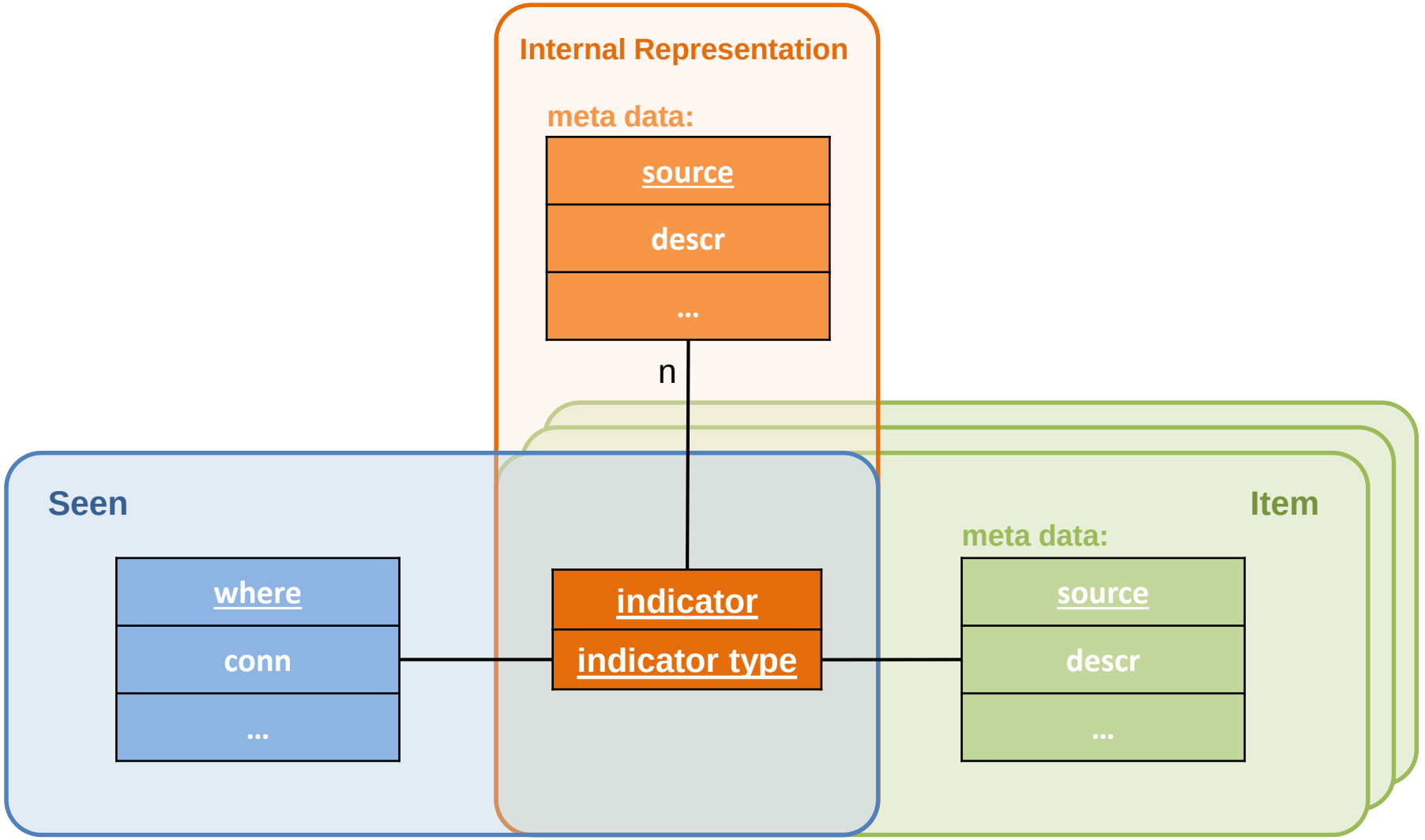
```
#field indicator indicator_type meta.source meta.desc meta.url
bro.org Intel::DOMAIN test-src1 new description    http://src1.example.com/id-x
bro.org Intel::DOMAIN test-src3 domain for testing http://src2.example.com/id-y
```



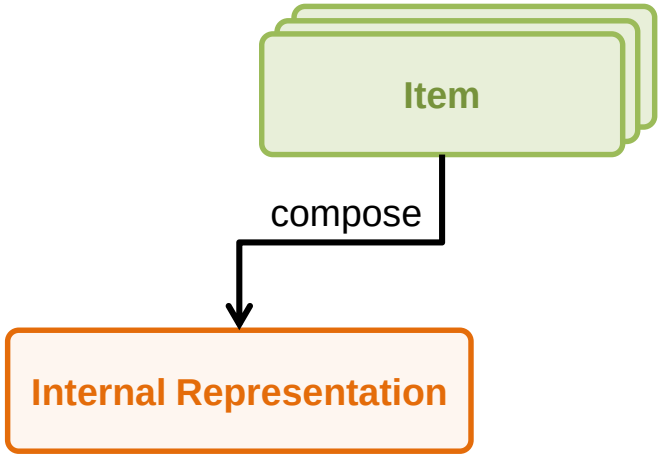
Matching Intel: Data Model



Matching Intel: Data Model



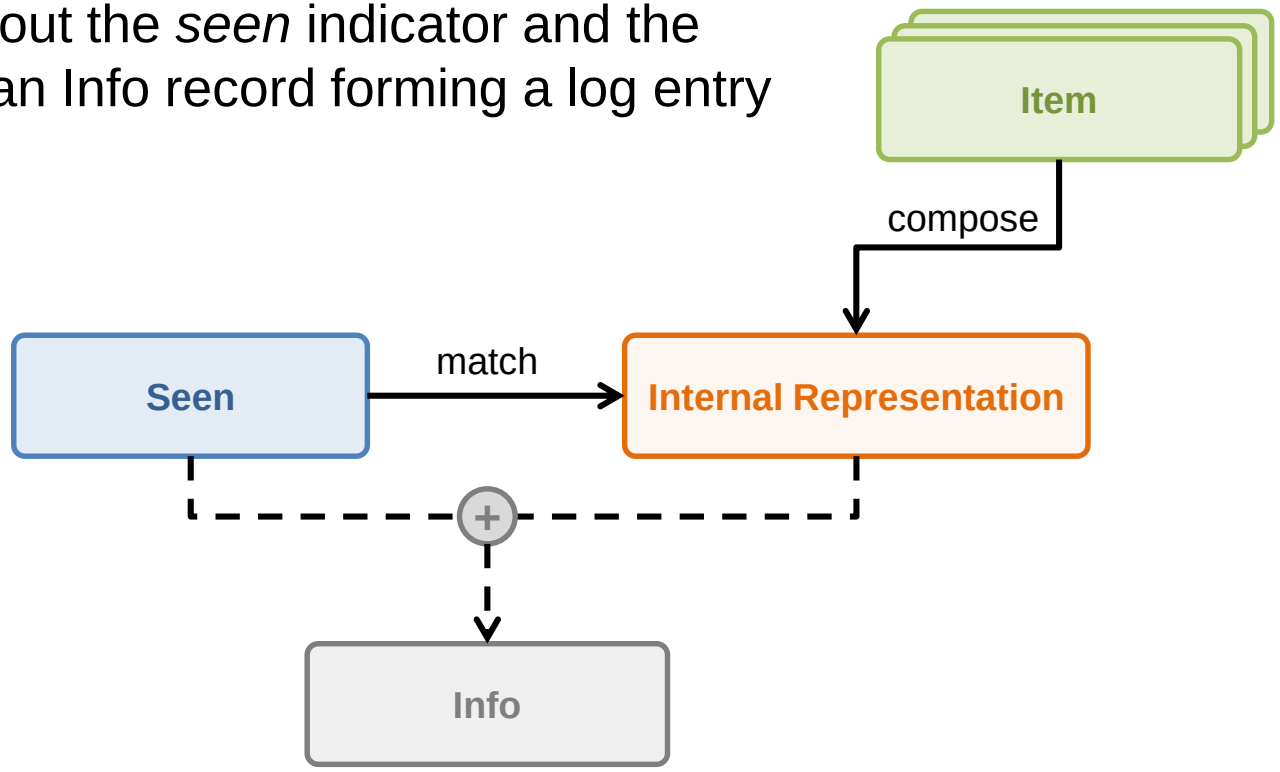
Matching Intel: Data Flow



Matching Intel: Data Flow

- **What you* *see* is what you match!**

- In case of a match, information about the *seen* indicator and the *matched* indicator is combined in an Info record forming a log entry



* you = Bro

Matching Intel: Seen

- **What you *see* is what you match:** `function Intel::seen(s: Intel::Seen)`
- Bro ships with a couple of observation-scripts in `frameworks/intel/seen`

dns.bro:

```
@load base/frameworks/intel
@load ./where-locations

event dns_request(c: connection, msg: dns_msg, query: string, qtype: count, qclass: count)
        {
        Intel::seen([$indicator=query,
                     $indicator_type=Intel::DOMAIN,
                     $conn=c,
                     $where=DNS::IN_REQUEST]);
}
```

Matching Intel: Seen

- **What you *see* is what you match:** `function Intel::seen(s: Intel::Seen)`
- Bro ships with a couple of observation-scripts in `frameworks/intel/seen`

dns.bro:

```
@load base/frameworks/intel
@load ./where-locations

event dns_request(c: connection, msg: dns_msg, query: string, qtype: count, qclass: count)
        {
        Intel::seen([$indicator=query,
                     $indicator_type=Intel::DOMAIN,
                     $conn=c,
                     $where=DNS::IN_REQUEST]);
}
```

frameworks/intel/seen/
where-locations.bro

Matching Intel: Seen

- **What you *see* is what you match:** `function Intel::seen(s: Intel::Seen)`
- Bro ships with a couple of observation-scripts in `frameworks/intel/seen`

dns.bro:

```
@load base/frameworks/intel
@load ./where-locations

event dns_request(c: connection, msg: dns_msg, query: string, qtype: count, qclass: count)
        {
        Intel::seen([$indicator=query,
                     $indicator_type=Intel::DOMAIN,
                     $conn=c,
                     $where=DNS::IN_REQUEST]);
}
```

frameworks/intel/seen/where-locations.bro

- The same type might be seen in various situations:
  - dns.bro
  - http-headers.bro
  - ssl.bro
  - x509.bro

  } Intel::DOMAIN

DSN Research Group
Institute of Telematics, Faculty of Informatics

Matching Intel: Seen

- **What you *see* is what you match:** `function Intel::seen(s: Intel::Seen)`
- Bro ships with a couple of observation-scripts in `frameworks/intel/seen`

conn-established.bro:

```
@load base/frameworks/intel
@load ./where-locations

event connection_established(c: connection)
        {
        if ( c$orig$state == TCP_ESTABLISHED &&
             c$resp$state == TCP_ESTABLISHED )
                {
                Intel::seen([$host=c$id$orig_h, $conn=c, $where=Conn::IN_ORIG]);
                Intel::seen([$host=c$id$resp_h, $conn=c, $where=Conn::IN_RESP]);
                }
}
```

Matching Intel: Seen

- **What you *see* is what you match:** `function Intel::seen(s: Intel::Seen)`
- Bro ships with a couple of observation-scripts in `frameworks/intel/seen`

conn-established.bro:

```
@load base/frameworks/intel
@load ./where-locations

event connection_established(c: connection)
        {
        if ( c$orig$state == TCP_ESTABLISHED &&
             c$resp$state == TCP_ESTABLISHED )
                {
                Intel::seen([$host=c$id$orig_h, $conn=c, $where=Conn::IN_ORIG]);
                Intel::seen([$host=c$id$resp_h, $conn=c, $where=Conn::IN_RESP]);
                }
}
```

Outline

- Basics
- Intel Framework Update
  - Type Intel::SUBNET
  - Suppressible Notices
  - Delete Intel
  - Extension Mechanism
  - Expiration of Intel Items
- Intel-Extensions
- Future Work

Outline

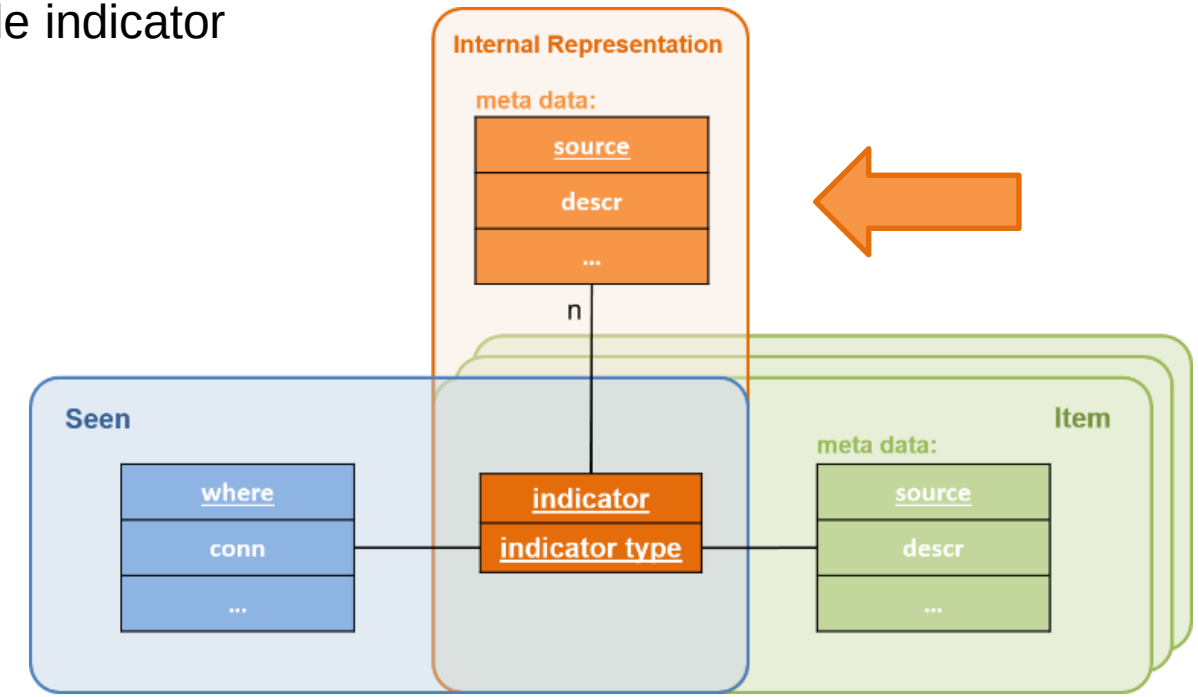
- Basics
- Intel Framework Update
  - Type Intel::SUBNET
  - Suppressible Notices
  - Delete Intel
  - Extension Mechanism
  - Expiration of Intel Items
- Intel-Extensions
- Future Work

Delete Intel

- `function Intel::remove (item: Intel::Item, purge_indicator: bool &default = F &optional)`

- Remove the corresponding meta data record (identified by `source`)
- If no meta data record left
- If `purge_indicator = T`

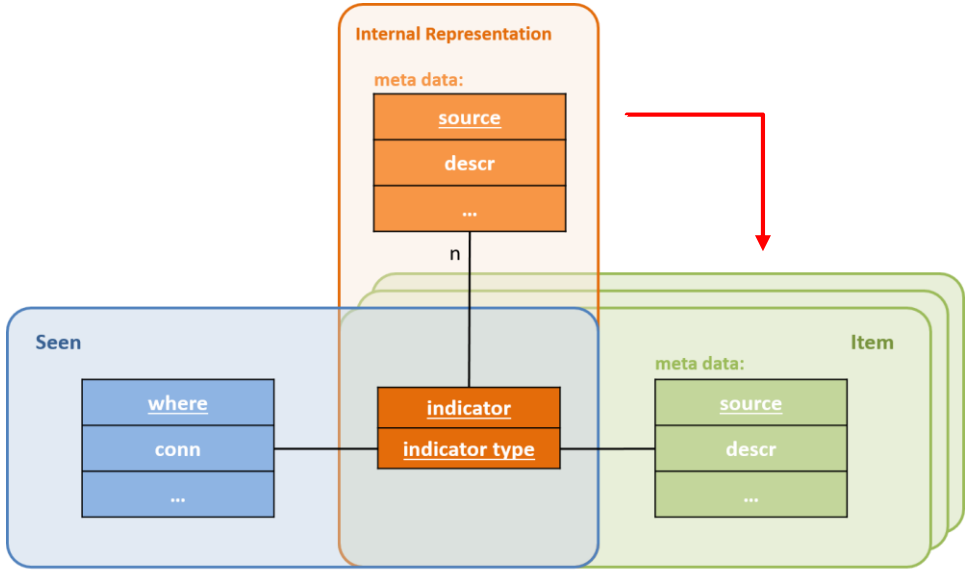
→ remove the whole indicator



DSN Research Group
Institute of Telematics, Faculty of Informatics

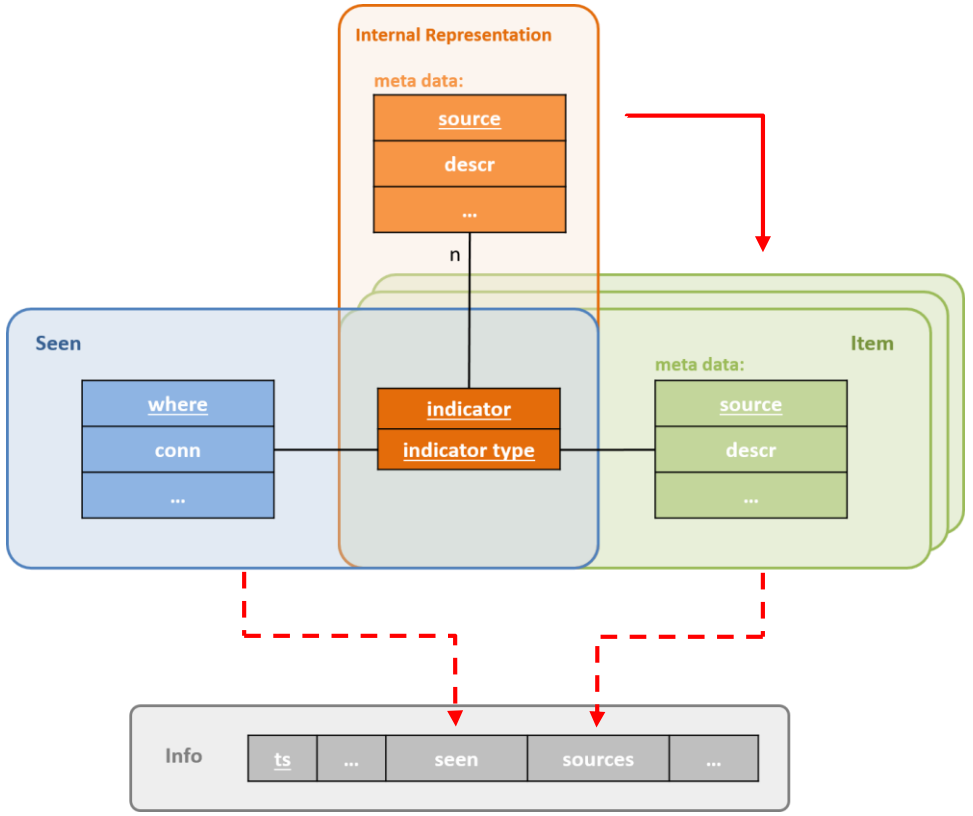
Extension Mechanism

- Based on Seth's Intel Extensions for Bro ≤ 2.4
- `hook Intel::extend_match(info: Intel::Info, s: Intel::Seen, items: set[Intel::Item]):bool`
  - `break`-ing the hook chain prevents logging
  - Manipulating the parameters influences what subsequent hook-handlers see
- Editing the `info` record the resulting log entry can be influenced
- "Aggregation Logic" to consider multiple matched items (usually multiple meta data records)



Extension Mechanism

- Based on Seth's Intel Extensions for Bro ≤ 2.4
- `hook Intel::extend_match(info: Intel::Info, s: Intel::Seen, items: set[Intel::Item]):bool`
  - `break`-ing the hook chain prevents logging
  - Manipulating the parameters influences what subsequent hook-handlers see
- Editing the `info` record the resulting log entry can be influenced
- "Aggregation Logic" to consider multiple matched items (usually multiple meta data records)



Extension Mechanism – Example

- Extend intel data with analyst name and log it

- Add `analyst` field to meta data record
- Add `analysts` field to info record for logging
- Propagate analyst name to logs using the `extend_match()` hook

example.bro:

```
module Intel;

export {
    redef record MetaData += {
        ## Analyst name.
        analyst: string &optional;
    };

    redef record Info += {
        ## Set of analysts involved.
        analysts: set[string] &optional &log;
    };
}

hook extend_match(info: Info, s: Seen, items: set[Item])
    {
    info$analysts = set();
    for ( item in items )
        {
        if ( item$meta?$analyst )
            add info$analysts[item$meta$analyst];
        }
    }
```


DSN Research Group
Institute of Telematics, Faculty of Informatics

Extension Mechanism – Example

- Extend intel data with analyst name and log it

- Add `analyst` field to meta data record
- Add `analysts` field to info record for logging
- Propagate analyst name to logs using the `extend_match()` hook

Aggregation Logic:
Insert all analyst names into a set.

example.bro:

```
module Intel;

export {
    redef record MetaData += {
        ## Analyst name.
        analyst: string &optional;
    };

    redef record Info += {
        ## Set of analysts involved.
        analysts: set[string] &optional &log;
    };
}

hook extend_match(info: Info, s: Seen, items: set[Item])
    {
    info$analysts = set();
    for ( item in items )
        {
        if ( item$meta?$analyst )
            add info$analysts[item$meta$analyst];
        }
    }
```

Extension Mechanism – Example

- Extend intel data with analyst name and log it

- Add `analyst` field to meta data record
- Add `analysts` field to info record for logging
- Propagate analyst name to logs using the `extend_match()` hook

- Other use cases:
  - File-related info
  - Whitelisting
  - …

Aggregation Logic:
Insert all analyst names into a set.

example.bro:

```
module Intel;

export {
    redef record MetaData += {
        ## Analyst name.
        analyst: string &optional;
    };

    redef record Info += {
        ## Set of analysts involved.
        analysts: set[string] &optional &log;
    };
}

hook extend_match(info: Info, s: Seen, items: set[Item])
    {
    info$analysts = set();
    for ( item in items )
        {
        if ( item$meta?$analyst )
            add info$analysts[item$meta$analyst];
        }
    }
```

Expiration of Intel Items

- Relevance of Intel data is of temporary nature (e.g. volatile cloud environments)

local.bro:

```
@load frameworks/intel/do_expire
redef Intel::item_expiration = 1hr;
```

- Reinsertion resets the expiration timeout

Expiration of Intel Items

- Relevance of Intel data is of temporary nature (e.g. volatile cloud environments)

local.bro:

```
@load frameworks/intel/do_expire
redef Intel::item_expiration = 1hr;
```

- Reinsertion resets the expiration timeout

Dosen’t work in Bro 2.5
Fixed in Bro 2.5.1
Thanks to Fatema & Seth!

Expiration of Intel Items

- Relevance of Intel data is of temporary nature (e.g. volatile cloud environments)

local.bro:

```
@load frameworks/intel/do_expire
redef Intel::item_expiration = 1hr;
```

- Reinsertion resets the expiration timeout

do_expire.bro:

```
hook item_expired(indicator: string, indicator_type: Type,
          metas: set[MetaData]) &priority=-10
          {
          # Trigger removal of the expired item:
          break;
          }
```

Usage:

- `item_expired` hook is called as soon as an indicator expires
- `break`-ing the chain of hooks the indicator get automatically purged
- On normal execution, the expiration interval is reset

Expiration of Intel Items

- Relevance of Intel data is of temporary nature (e.g. volatile cloud environments)

local.bro:

```
@load frameworks/intel/do_expire
redef Intel::item_expiration = 1hr;
```

- Reinsertion resets the expiration timeout

do_expire.bro:

```
hook item_expired(indicator: string, indicator_type: Type,
          metas: set[MetaData]) &priority=-10
          {
          # Trigger removal of the expired item:
          break;
          }
```

Usage:

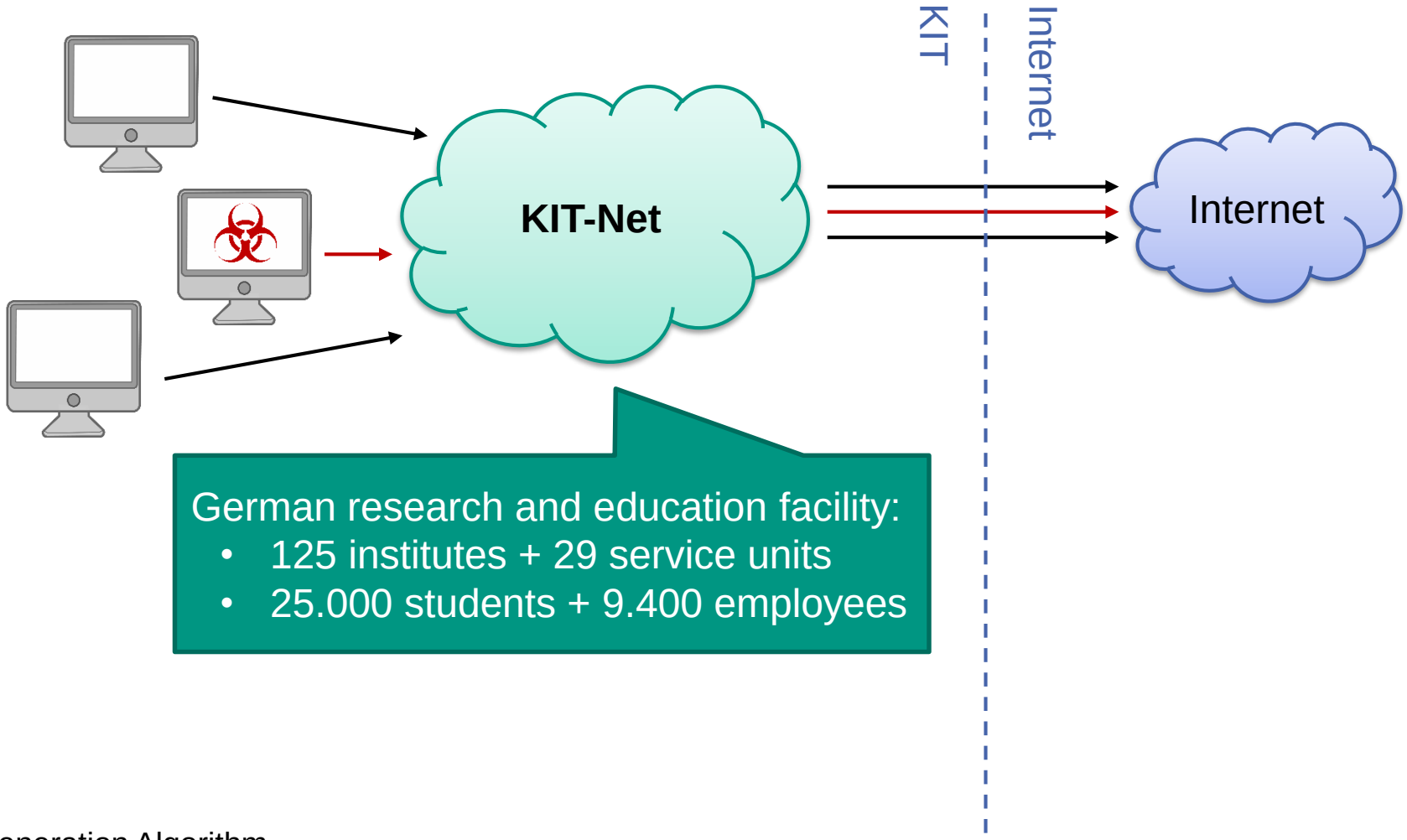
- `item_expired` hook is called as soon as an indicator expires
- `break`-ing the chain of hooks the indicator get automatically purged
- On normal execution, the expiration interval is reset

Once an intel file changes, **all** items are reinserted
→ Keep Intel files and “in-memory DB” in sync

Outline

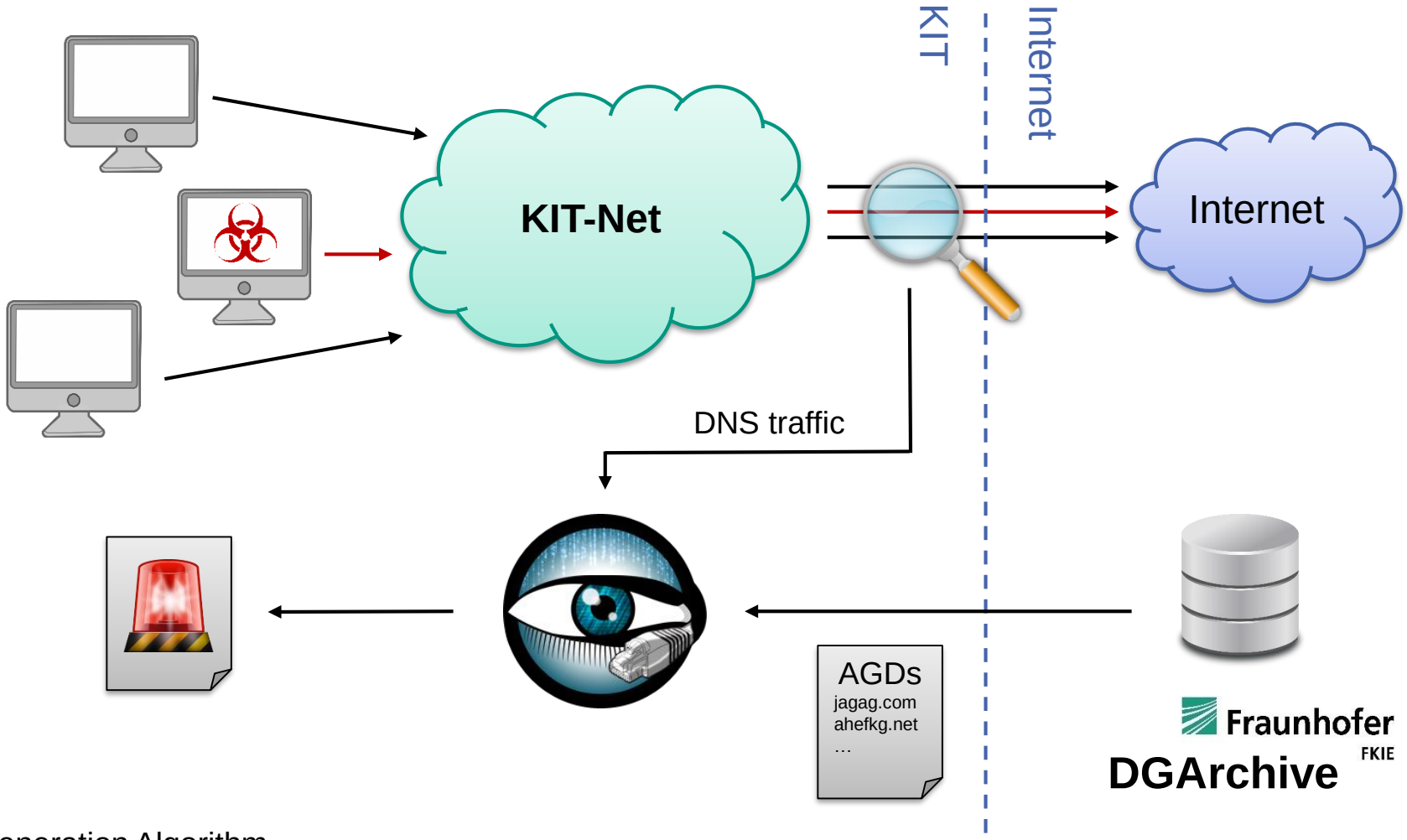
- Basics
- Intel Framework Update
- Intel-Extensions
  - Per Item Expiration
  - Remote Control
- Future Work

DGA Scenario



DGA = Domain Generation Algorithm

DGA Scenario



DGA = Domain Generation Algorithm
AGD = Algorithmically Generated Domain

Bringing it all together…

- General scenario: Intel (obtained from multiple feeds) with different characteristics
- Differences in quality (reliability, durability, …) = different expiration intervals

- So far: Keeping intel files and in-memory DB “in sync”
- Different expiration intervals require external logic

- Idea: Support individual expiration intervals per item
- Implementation: Per-item expiration as combining
  - Intel Expiration and
  - Intel Extensions

github.com/J-Gras/intel-extensions

Per-Item Expiration

- Installation: `bro-pkg install intel-extensions`

- Usage: `@load intel-extensions/scripts/item_expire`
- Add `meta.expire` column to the corresponding intel files

item_expire.bro:

```
export {
        # Default expiration interval for single intelligence items.
        const default_per_item_expiration = -1 min &redef;

        redef record MetaData += {
                # Expiration interval of the intelligence item.
                expire:     interval &default=default_per_item_expiration;

                # Internal: Timestamp of last hit.
                last_match: time     &default=network_time();
        };

        global single_item_expired: hook(item: Item);
}
```

Per-Item Expiration: Implementation

- `hook extend_match(info: Info, s: Seen, items: set[Item])`
- Prevent logging of already expired items:
  - Check: `last_match + expire < network_time`
    `single_item_expired()` hook to allow recovery
    } item already expired?
  - If expired: Remove item
  - Otherwise: Update `last_match`
  - Aggregation Logic: Generate a hit if there is **any** valid match

Per-Item Expiration: Implementation

- `hook extend_match(info: Info, s: Seen, items: set[Item])`
- Prevent logging of already expired items:
  - Check: `last_match + expire < network_time` } item already expired?
    `single_item_expired()` hook to allow recovery
  - If expired: Remove item
  - Otherwise: Update `last_match`
  - Aggregation Logic: Generate a hit if there is **any** valid match

- `hook Intel::item_expired(indicator: string, indicator_type: Type, metas: set[MetaData])`
- Use global expiration interval for “garbage collection”
  - Check whether the items already expired
  - If expired: Remove item

Remote Control

- Problem: Intel files and “in-memory” DB might differ
  → hard to manage/debug
- Solution: Python script for simple for basic management tasks using broker

```
intel-mgr.py [-p PORT] [-a IP] OPERATION INDICATOR TYPE
```

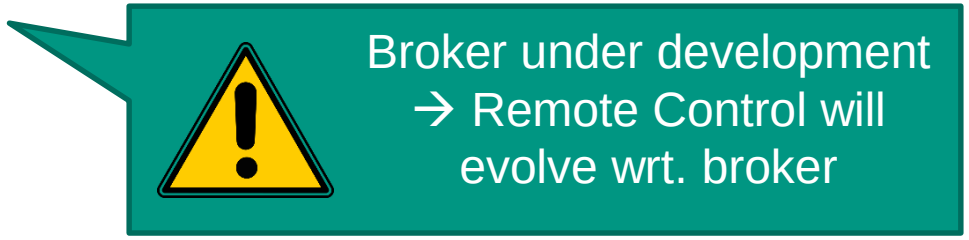
- Supported operations:
  - query → yes/no answer
  - remove → fire and forget
  - insert → uses default metadata

Remote Control

- Problem: Intel files and “in-memory” DB might differ
  → hard to manage/debug
- Solution: Python script for simple for basic management tasks using broker

```
intel-mgr.py [-p PORT] [-a IP] OPERATION INDICATOR TYPE
```

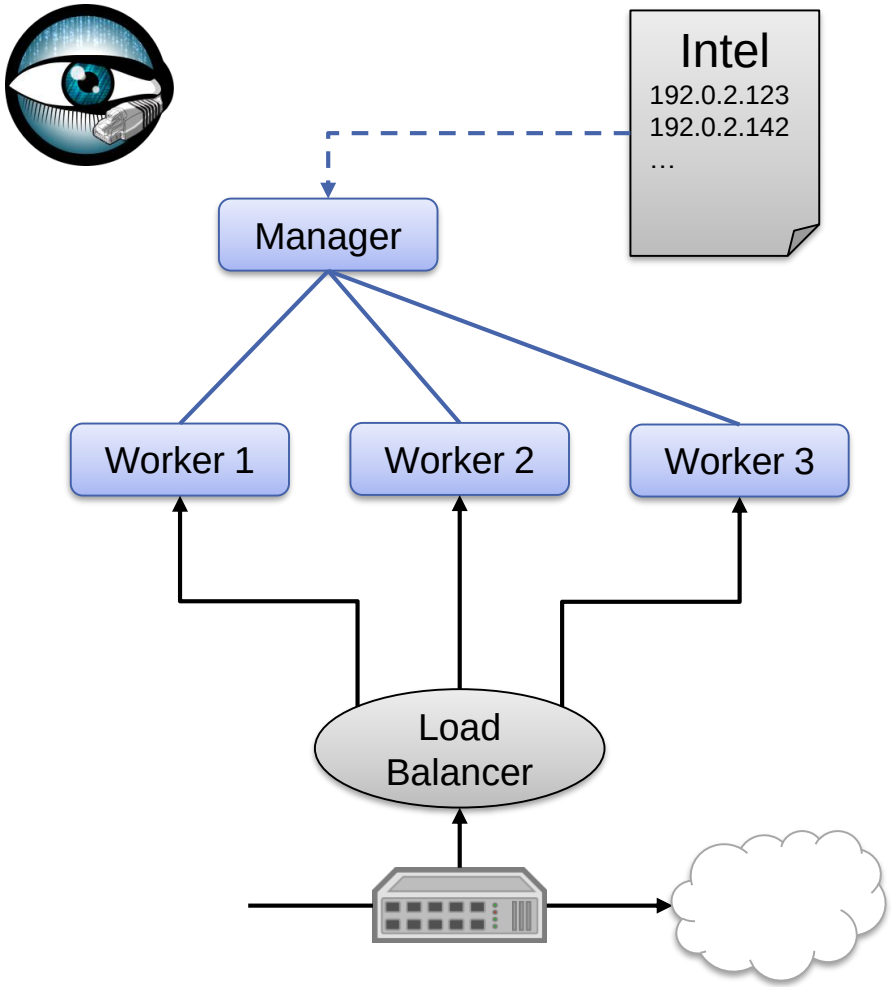
- Supported operations:
  - query → yes/no answer
  - remove → fire and forget
  - insert → uses default metadata



Outline

- Basics
- Intel Framework Update
- Intel-Extensions
- Future Work

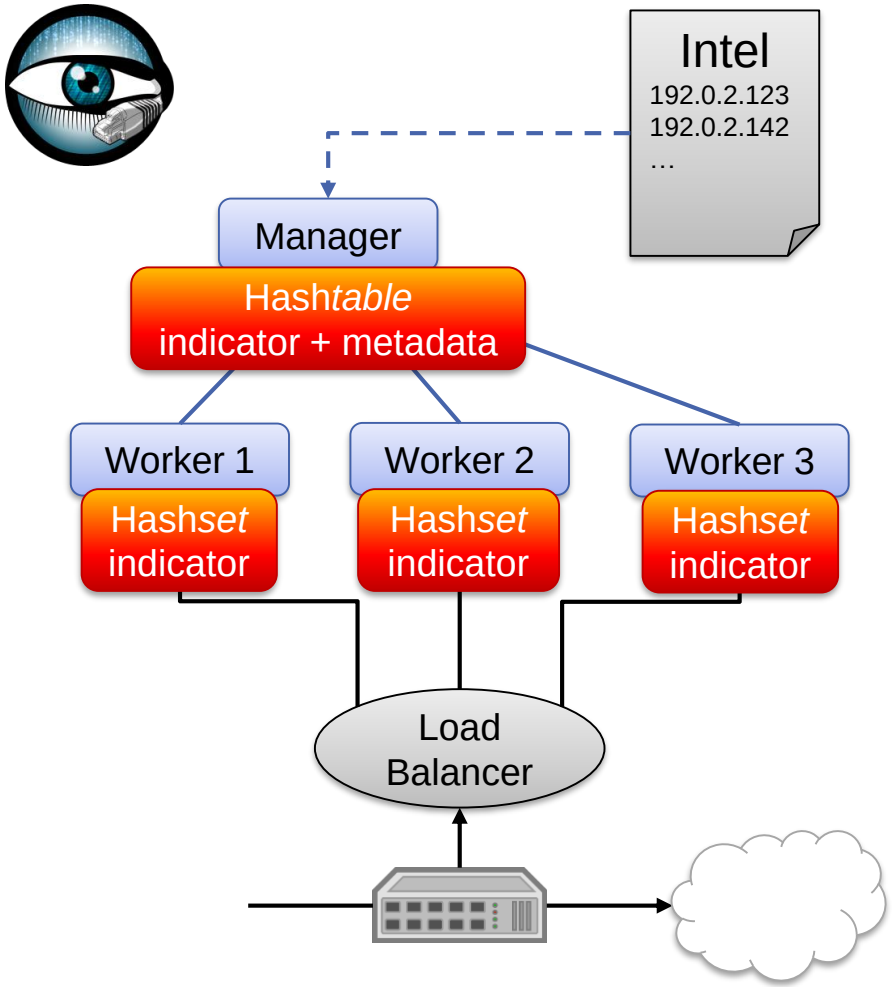
Improve Intel Matching



- Bro supports clustering to scale
  - Manager → coordination
  - Worker → traffic processing
- Intelligence Framework
  - Manages indicator + metadata
  - Distributes indicators for matching
- Requirements:
  - Efficient matching
  - Support removal
- Idea: Use Cuckoo Filter?

© Matthias Vallentin

Improve Intel Matching



- Bro supports clustering to scale
  - Manager → coordination
  - Worker → traffic processing
- Intelligence Framework
  - Manages indicator + metadata
  - Distributes indicators for matching

- Requirements:
  - Efficient matching
  - Support removal
- Idea: Use Cuckoo Filter?

© Matthias Vallentin

Preliminary Results

- Hashset/-table:
  - 0.8 µs/operation
  - no false-positives
  - ▶ ≥150 byte/indicator
- Cuckoo Filter
  - 0.4 µs/operation
  - 0.1% false-positive ratio
  - ▶ 3 byte/indicator

Preliminary Results

- Hashset/-table:
  - 0.8 µs/operation
  - no false-positives
  - ▶ ≥150 byte/indicator
- Cuckoo Filter
  - 0.4 µs/operation
  - 0.1% false-positive ratio
  - ▶ 3 byte/indicator

Number of indicators:	Hashtable	Hashset	Cuckoo Filter
1 m	200 MiB	150 MiB	3 MiB
16 m	3.2 GiB	2.2 GiB	48 MiB

Future Work

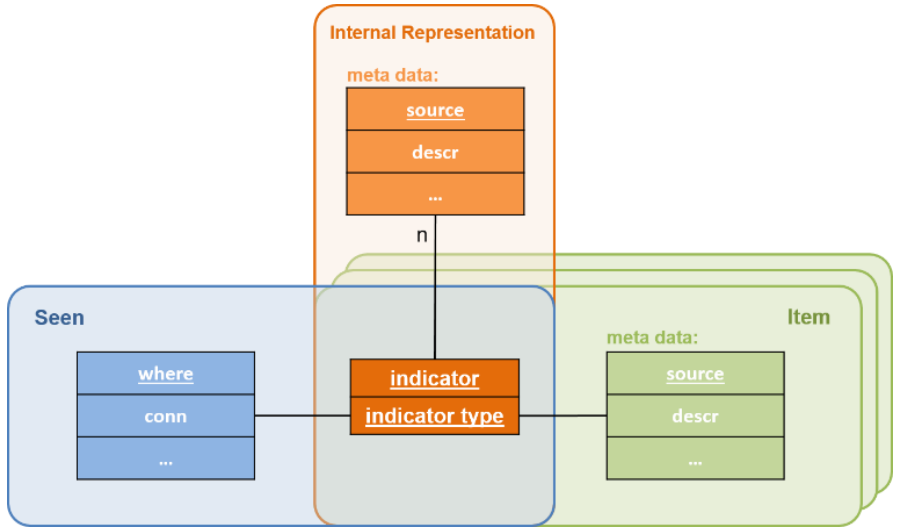
- Estimate typical Intel Framework workloads

- Add Rust implementation of Cuckoo Filter
- Implement Cuckoo Filter in C++?
- ▶ Compare Implementations

- Setup clustered test-environment
- Observe effects of false-positives
- ▶ “Real-life” benchmarks

Summary

- Basics
  - Data Model
  - Ingesting Intel
  - Matching Intel
- Intel Framework Update
  - Delete Intel
  - Extension Mechanism
  - Expiration of Intel Items
- Intel-Extensions
  - Per Item Expiration
  - Remote Control
- Future Work




```
@load frameworks/intel/seen
@load frameworks/intel/do_notice

const feed_directory = “/usr/local/bro/feeds”;

redef Intel::read_files += {
          feed_directorty + “/example.intel”,
          #...
};
```

ADDITIONAL SLIDES

Intel::SUBNET

- Patricia-Trie to match IPs to given subnets in $\mathcal{O}(\log(n))$
- seen IP → match exactly vs. belonging to a subnet

example.intel:

```
#fields indicator       indicator_type  meta.source  meta.desc    meta.url
192.168.1.1             Intel::ADDR     source1      test ip      http://src1.example.com/id-1
192.168.2.0/24          Intel::SUBNET   source1      test subnet  http://src1.example.com/id-2
192.168.142.1           Intel::ADDR     source1      test ip      http://src1.example.com/id-3
192.168.142.0/24        Intel::SUBNET   source1      test subnet  http://src1.example.com/id-4
```

Intel::SUBNET

- Patricia-Trie to match IPs to given subnets in $\mathcal{O}(\log(n))$
- seen IP → match exactly vs. belonging to a subnet

example.intel:

```
#fields indicator       indicator_type  meta.source  meta.desc    meta.url
192.168.1.1             Intel::ADDR     source1      test ip      http://src1.example.com/id-1
192.168.2.0/24          Intel::SUBNET   source1      test subnet  http://src1.example.com/id-2
192.168.142.1           Intel::ADDR     source1      test ip      http://src1.example.com/id-3
192.168.142.0/24        Intel::SUBNET   source1      test subnet  http://src1.example.com/id-4
```

intel.log:

```
#fields  […] seen.indicator  seen.indicator_type  seen.where  matched                    sources
#types   […] string          enum                 enum        set[enum]                  set[string]
         […] 192.168.1.1     Intel::ADDR          SOMEWHERE   Intel::ADDR                source1
         […] 192.168.2.1     Intel::ADDR          SOMEWHERE   Intel::SUBNET              source1
         […] 192.168.142.1   Intel::ADDR          SOMEWHERE   Intel::SUBNET,Intel::ADDR  source1
```

Suppressible Intel Notices

- “The notice framework supports suppression for notices if the author of the script that is generating the notice has indicated to the notice framework how to identify notices that are intrinsically the same.” – Bro Documentation

- Default script shipped with Bro: `@load frameworks/intel/do_notice`
- Intel Notice is identified by Indicator and connection endpoints
  - {indicator, orig_h, resp_h}

- Also new: More information for notice mails
  - service
  - source

- Feel free to customize!

Extension Mechanism – Example: Whitelisting

- Whitelisting to prevent erroneous intel hits (e.g. on your own infrastructure)
- Implementation: 1. Add metadata field `whitelist: bool`
  2. Prevent logging if `whitelist = T`

whitelist.bro:

```
hook Intel::extend_match(info: Info, s: Seen, items:
           set[Item]) &priority=9
           {
           local whitelisted = F;
           for ( item in items )
                      {
                      if ( item$meta$whitelist )
                                 {
                                 whitelisted = T;
                                 break;
                                 }
                      }

           if ( whitelisted )
                      # Prevent logging
                      break;
           }
```

- Usage:

```
@load frameworks/intel/whitelist
```

- Add `meta.whitelist` column to the corresponding intel file

Extension Mechanism – Example: Whitelisting

- Whitelisting to prevent erroneous intel hits (e.g. on your own infrastructure)
- Implementation: 1. Add metadata field `whitelist: bool`
  2. Prevent logging if `whitelist = T`

whitelist.bro:

```
hook Intel::extend_match(info: Info, s: Seen, items:
            set[Item]) &priority=9
            {
            local whitelisted = F;
            for ( item in items )
                        {
                        if ( item$meta$whitelist )
                                    {
                                    whitelisted = T;
                                    break;
                                    }
                        }

            if ( whitelisted )
                        # Prevent logging
                        break;
            }
```

Aggregation Logic:
If there is a single whitelisted instance, then ignore the hit.

- Usage:

```
@load frameworks/intel/whitelist
```

- Add `meta.whitelist` column to the corresponding intel file