MISP and Decaying of Indicators

An indicator scoring method and ongoing imple-

Team CIRCL

info@circl.lu

August 5, 2022

Expiring IOCs: Why and How?

Expiring IOCs: Why and How?

Indicators - Problem Statement

- **Sharing information** about threats **is crucial**
- Organisations are sharing more and more

Contribution by **unique organisation** (`Orgc.name`) on MISPPriv:

Date	Unique Org
2013	17
2014	43
2015	82
2016	105
2017	118
2018	125
2019-10	135

```
{
    "distribution": [1, 2, 3]
}
```


Indicators - Problem Statement
Sharing information about threats is crucial
Organisations are sharing more and more
Contribution by unique organisation (Orgc.name) on MISPPriv:
Date Unique Org
2013 17
2014 43
2015 82
2016 105
2017 118
2018 125
2019-10 135
{ "distribution": [1, 2, 3] }

Indicators - Problem Statement

- Various users and organisations can share data via MISP, multiple parties can be involved
  - **Trust**, **data quality** and **time-to-live** issues
  - Each user/organisation has **different use-cases** and interests
    - Conflicting interests such as operational security, attribution,... (depends on the user)

→ Can be partially solved with *Taxonomies*

Indicators - Problem Statement
Various users and organisations can share data via MISP, multiple parties can be involved
Trust, data quality and time-to-live issues
Each user/organisation has different use-cases and interests
Conflicting interests such as operational security, attribution,... (depends on the user)
→ Can be partially solved with Taxonomies

Indicators - Problem Statement

- Various users and organisations can share data via MISP, multiple parties can be involved
  - **Trust**, **data quality** and **time-to-live** issues
  - Each user/organisation has **different use-cases** and interests
    - Conflicting interests such as operational security, attribution,... (depends on the user)

  → Can be partially solved with *Taxonomies*

- Attributes can be shared in large quantities (more than 7.3 million on MISPPRIV)
  - Partial info about their **freshness** (*Sightings*)
  - Partial info about their **validity** (last update)

  → Can be partially solved with our *Decaying model*

Indicators - Problem Statement
Various users and organisations can share data via MISP, multiple parties can be involved
Trust, data quality and time-to-live issues
Each user/organisation has different use-cases and interests
Conflicting interests such as operational security, attribution,... (depends on the user)
→ Can be partially solved with Taxonomies
Attributes can be shared in large quantities (more than 7.3 million on MISPPRIV)
Partial info about their freshness (Sightings)
Partial info about their validity (last update)
→ Can be partially solved with our Decaying model

Requirements to enjoy the decaying feature in MISP

- Starting from **MISP 2.4.116**, the decaying feature is available
- Don't forget to update the decay models and enable the ones you want
- The decaying feature has no impact on the information in MISP, it's just an overlay to be used in the user-interface and API
- Decay strongly relies on *Taxonomies* and *Sightings*, don't forget to review their configuration

Requirements to enjoy the decaying feature in MISP

- Starting from **MISP 2.4.116**, the decaying feature is available
- Don't forget to update the decay models and enable the ones you want
- The decaying feature has no impact on the information in MISP, it's just an overlay to be used in the user-interface and API
- Decay strongly relies on *Taxonomies* and *Sightings*, don't forget to review their configuration

Sightings - Refresher

Sightings add temporal context to indicators. A user, script or an IDS can extend the information related to indicators by reporting back to MISP that an indicator has been `seen`, or that an indicator can be considered as a `false-positive`

- *Sightings* give more credibility/visibility to indicators
- This information can be used to **prioritise and decay indicators**

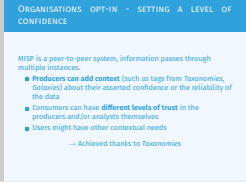
Sightings - Refresher
Sightings add temporal context to indicators. A user, script or an IDS can extend the information related to indicators by reporting back to MISP that an indicator has been seen, or that an indicator can be considered as a false-positive
Sightings give more credibility/visibility to indicators
This information can be used to prioritise and decay indicators
(44/0/0)

Organisations opt-in - setting a level of confidence

MISP is a peer-to-peer system, information passes through multiple instances.

- **Producers can add context** (such as tags from *Taxonomies*, *Galaxies*) about their asserted confidence or the reliability of the data
- Consumers can have **different levels of trust** in the producers and/or analysts themselves
- Users might have other contextual needs

→ Achieved thanks to *Taxonomies*



Taxonomies - Refresher (1)

Taxonomies

« previous | 1 | 2 | next »

Id ↑	Namespace	Description	Version	Enabled	Required	Active Tags	Actions
181	workflow	Workflow support language is a common language to support intelligence analysts to perform their analysis on data and information.	9	Yes	☐	**27** / 26 (enable all)	
180	vocabulaire-des-probabilites-estimatives	Ce vocabulaire attribue des valeurs en pourcentage à certains énoncés de probabilité	2	Yes	☐	**5** / 5	
179	threats-to-dns	An overview of some of the known attacks related to DNS as described by Torabi, S., Boukhtouta, A., Assi, C., & Debbabi, M. (2018) in Detecting Internet Abuse by Analyzing Passive DNS Traffic: A Survey of Implemented Systems. IEEE Communications Surveys & Tutorials, 1–1. doi:10.1109/comst.2018.2849614	1	No	☐	**0** / 18	
178	targeted-threat-index	The Targeted Threat Index is a metric for assigning an overall threat ranking score to email messages that deliver malware to a victim's computer. The TTI metric was first introduced at SecTor 2013 by Seth Hardy as part of the talk "RATastrophe: Monitoring a Malware Menagerie" along with Katie Kleemola and Greg Wiseman.	2	Yes	☐	**11** / 11	

- Tagging is a simple way to attach a classification to an *Event* or an *Attribute*
- Classification must be globally used to be efficient

Taxonomies - Refresher (1)
- Tagging is a simple way to attach a classification to an *Event* or an *Attribute*
- Classification must be globally used to be efficient

Taxonomies - Refresher (2)

ADMIRALTY-SCALE Taxonomy Library

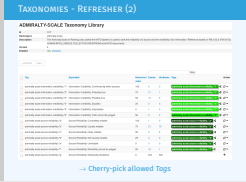
Id	127
Namespace	admiralty-scale
Description	The Admiralty Scale or Ranking (also called the NATO System) is used to rank the reliability of a source and the credibility of an information. Reference based on FM 2-22.3 (FM 34-52) HUMAN INTELLIGENCE COLLECTOR OPERATIONS and NATO documents.
Version	4
Enabled	Yes (disable)

« previous next »

Filter

Tag	Expanded	Numerical value	Events	Attributes	Tags	Action
admiralty-scale:information-credibility="1"	Information Credibility: Confirmed by other sources	100	6	0	admiralty-scale:information-credibility="1"	
admiralty-scale:information-credibility="2"	Information Credibility: Probably true	75	21	1	admiralty-scale:information-credibility="2"	
admiralty-scale:information-credibility="3"	Information Credibility: Possibly true	50	16	5	admiralty-scale:information-credibility="3"	
admiralty-scale:information-credibility="4"	Information Credibility: Doubtful	25	2	0	admiralty-scale:information-credibility="4"	
admiralty-scale:information-credibility="5"	Information Credibility: Improbable	0	1	0	admiralty-scale:information-credibility="5"	
admiralty-scale:information-credibility="6"	Information Credibility: Truth cannot be judged	50	9	2	admiralty-scale:information-credibility="6"	
admiralty-scale:source-reliability="a"	Source Reliability: Completely reliable	100	1	0	admiralty-scale:source-reliability="a"	
admiralty-scale:source-reliability="b"	Source Reliability: Usually reliable	75	21	76	admiralty-scale:source-reliability="b"	
admiralty-scale:source-reliability="c"	Source Reliability: Fairly reliable	50	9	8	admiralty-scale:source-reliability="c"	
admiralty-scale:source-reliability="d"	Source Reliability: Not usually reliable	25	2	0	admiralty-scale:source-reliability="d"	
admiralty-scale:source-reliability="e"	Source Reliability: Unreliable	0	0	0	admiralty-scale:source-reliability="e"	
admiralty-scale:source-reliability="f"	Source Reliability: Reliability cannot be judged	50	10	7	admiralty-scale:source-reliability="f"	
admiralty-scale:source-reliability="g"	Source Reliability: Deliberatly deceptive	0	N/A	N/A		

→ Cherry-pick allowed *Tags*



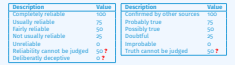
Taxonomies - Refresher (3)

- Some taxonomies have `numerical_value`
  → Can be used to prioritise *Attributes*

Description	Value
Completely reliable	100
Usually reliable	75
Fairly reliable	50
Not usually reliable	25
Unreliable	0
Reliability cannot be judged	50 ?
Deliberatly deceptive	0 ?

Description	Value
Confirmed by other sources	100
Probably true	75
Possibly true	50
Doubtful	25
Improbable	0
Truth cannot be judged	50 ?

Taxonomies - Refresher (3)
Some taxonomies have numerical_value
→ Can be used to prioritise Attributes



Scoring Indicators: Our solution

$$\texttt{score}(\texttt{Attribute}) = \texttt{base\_score}(\texttt{Attribute}, \texttt{Model}) \bullet \texttt{decay}(\texttt{Model}, \texttt{time})$$

Where,

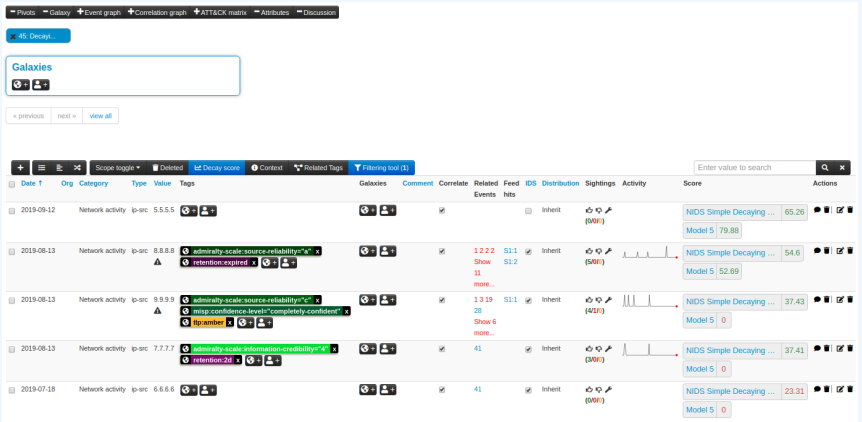
- $\texttt{score} \in [0, +\infty$
- $\texttt{base\_score} \in [0, 100]$
- `decay` is a function defined by model's parameters controlling decay speed
- `Attribute` Contains *Attribute*'s values and metadata (*Taxonomies*, *Galaxies*, ...)
- `Model` Contains the *Model*'s configuration

Scoring Indicators: Our solution
score(Attribute) = base_score(Attribute, Model) • decay(Model, time)
Where,
score ∈ [0, +∞
base_score ∈ [0, 100]
decay is a function defined by model's parameters controlling decay speed
Attribute Contains Attribute's values and metadata (Taxonomies, Galaxies, ...)
Model Contains the Model's configuration

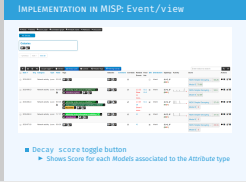
Current implementation in MISP

Current implementation in MISP

Implementation in MISP: Event/view



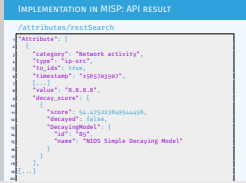
- Decay score toggle button
  - Shows Score for each *Models* associated to the *Attribute* type



Implementation in MISP: API result

/attributes/restSearch

```
"Attribute": [
  {
    "category": "Network activity",
    "type": "ip-src",
    "to_ids": true,
    "timestamp": "1565703507",
    [...]
    "value": "8.8.8.8",
    "decay_score": [
      {
        "score": 54.475223849544456,
        "decayed": false,
        "DecayingModel": {
          "id": "85",
          "name": "NIDS Simple Decaying Model"
        }
      }
    ],
[...]
```

Implementation in MISP: Playing with Models

- **Automatic scoring** based on default values
- **User-friendly UI** to manually set *Model* configuration (lifetime, decay, etc.)
- **Simulation** tool
- Interaction through the **API**
- Opportunity to create your **own** formula or algorithm

Implementation in MISP: Playing with Models
- **Automatic scoring** based on default values
- **User-friendly UI** to manually set *Model* configuration (lifetime, decay, etc.)
- **Simulation** tool
- Interaction through the **API**
- Opportunity to create your **own** formula or algorithm

Decaying Models in Depth

Scoring Indicators: base_score (1)

$\texttt{score}(\texttt{Attribute}) = \texttt{base\_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$

When scoring indicators[1], multiple parameters[2] can be taken into account. The **base score** is calculated with the following in mind:

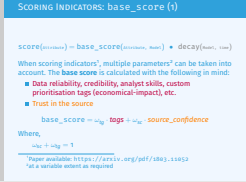
- Data reliability, credibility, analyst skills, custom prioritisation tags (economical-impact), etc.
- Trust in the source

$$\texttt{base\_score} = \omega_{tg} \cdot \textit{tags} + \omega_{sc} \cdot \textit{source\_confidence}$$

Where,

$$\omega_{sc} + \omega_{tg} = 1$$

[1] Paper available: https://arxiv.org/pdf/1803.11052
[2] at a variable extent as required



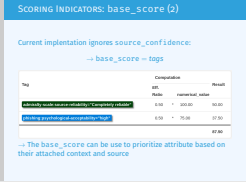
Scoring Indicators: base_score (2)

Current implentation ignores source_confidence:

$\rightarrow$ base_score = *tags*

Tag	Computation			Result
	Eff. Ratio		numerical_value	
admiralty-scale:source-reliability="Completely reliable"	0.50	*	100.00	50.00
phishing:psychological-acceptability="high"	0.50	*	75.00	37.50
				87.50

$\rightarrow$ The base_score can be use to prioritize attribute based on their attached context and source



Scoring Indicators: decay speed (1)

$\texttt{score}(\texttt{Attribute}) = \texttt{base\_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$

The decay is calculated using:

- The lifetime of the indicator
  - May vary depending on the indicator type
  - short for an IP, long for an hash
- The decay rate, or speed at which an attribute loses score over time
- The time elapsed since the latest update or sighting

Scoring Indicators: decay speed (1)
score(Attribute) = base_score(Attribute, Model) • decay(Model, time)
The decay is calculated using:
The lifetime of the indicator
May vary depending on the indicator type
short for an IP, long for an hash
The decay rate, or speed at which an attribute loses score over time
The time elapsed since the latest update or sighting

Scoring Indicators: putting it all toghether

→ `decay rate` is **re-initialized upon sighting** addition, or said differently, the `score` is reset to its base score as new *sightings* are applied.

$$score = base\_score \cdot \left(1 - \left(\frac{t}{\tau}\right)^{\frac{1}{\delta}}\right)$$

- $\tau =$ `lifetime`
- $\delta =$ `decay speed`

Scoring Indicators: putting it all toghether
→ decay rate is re-initialized upon sighting addition, or said differently, the score is reset to its base score as new sightings are applied.
score = base_score · (1 − (t/τ)^(1/δ))
τ = lifetime
δ = decay speed

Implementation in MISP: Models definition

$$\overrightarrow{score} = base\_score \cdot \left(1 - \left(\frac{t}{\tau}\right)^{\frac{1}{\delta}}\right)$$

Models are an instanciation of the formula where elements can be defined:

- Parameters: `lifetime, decay_rate, threshold`
- `base_score`
- `default base_score`
- formula
- associate *Attribute* types
- creator organisation

Implementation in MISP: Models definition
Models are an instanciation of the formula where elements can be defined:
Parameters: lifetime, decay_rate, threshold
base_score
default base_score
formula
associate Attribute types
creator organisation

Implementation in MISP: Models Types

Multiple model types are available

- **Default Models**: Models created and shared by the community. Available from `misp-decaying-models` repository[3].
  - → Not editable
- **Organisation Models**: Models created by a user belonging to an organisation
  - These models can be hidden or shared to other organisation
  - → Editable

[3] `https://github.com/MISP/misp-decaying-models.git`

Implementation in MISP: Models Types
Multiple model types are available
Default Models: Models created and shared by the community. Available from misp-decaying-models repository[3].
→ Not editable
Organisation Models: Models created by a user belonging to an organisation
These models can be hidden or shared to other organisation
→ Editable
[3] https://github.com/MISP/misp-decaying-models.git

Implementation in MISP: Index

Decaying Models

« previous | next »

All Models | My Models | Shared Models | Default Models

ID	Organization	Usable to everyone	Name	Description	Parameters { }	Formula	# Assigned Types	Version	Enabled	Actions
29	1	✔	Phishing model	Simple model to rapidly decay phishing website.	{ "lifetime": 3, "decay_speed": 2.3, "threshold": 30, "default_base_score": 80, "base_score_config": { "estimative-language": 0.5, "phishing": 0.5 } }	Polynomial	9	1	✔	
85	1	✖	NIDS Simple Decaying Model MISP	Simple decaying model for Network Intrusion Detection System (NIDS).	{ "lifetime": 120, "decay_speed": 2, "threshold": 30, "default_base_score": 80, "base_score_config": { "estimative-language": 0.25, "priority-level": 0.25, "retention": 0.25, "targeted-threat-index": 0.125, "false-positive": 0.125 } }	Polynomial	13	1	✔	

Page 1 of 1, showing 2 records out of 2 total, starting on record 1, ending on 2

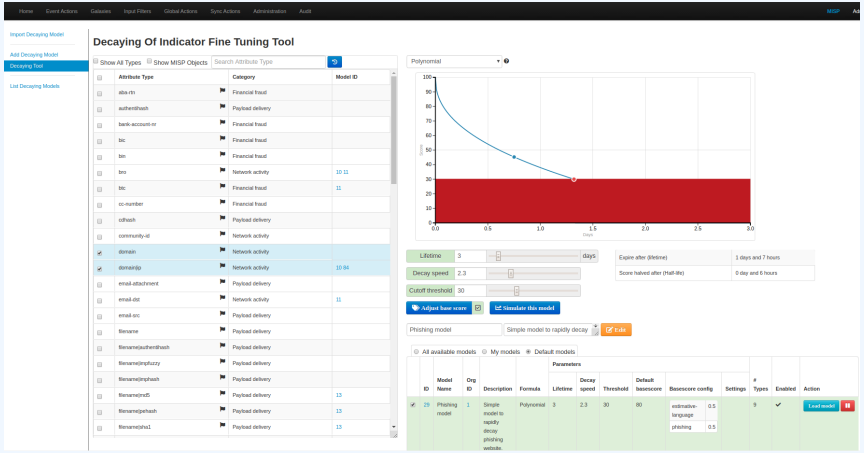
« previous | next »

View, update, add, create, delete, enable, export, import

Implementation in MISP: Index

View, update, add, create, delete, enable, export, import

Implementation in MISP: Fine tuning tool

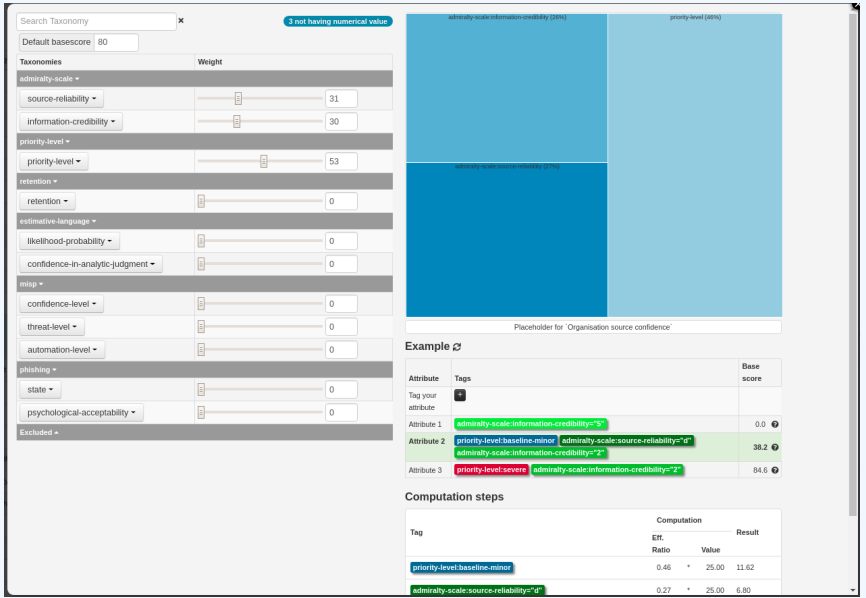


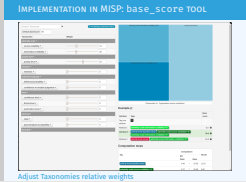
Create, modify, visualise, perform mapping

Implementation in MISP: Fine tuning tool

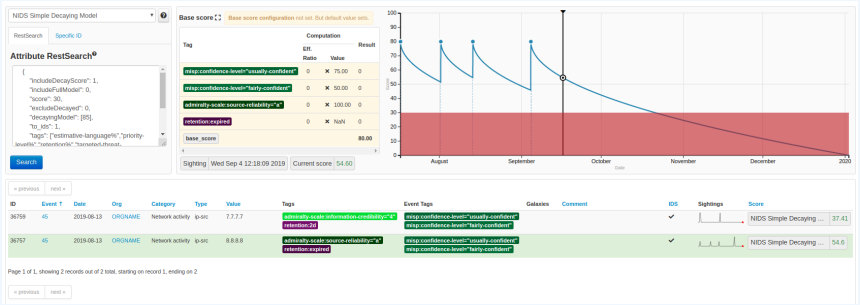
Create, modify, visualise, perform mapping

Implementation in MISP: base_score tool





Implementation in MISP: simulation tool



Simulate *Attributes* with different *Models*

Implementation in MISP: simulation tool

Simulate Attributes with different Models

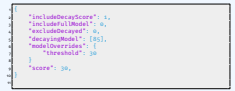
Implementation in MISP: API query body

/attributes/restSearch

```
{
    "includeDecayScore": 1,
    "includeFullModel": 0,
    "excludeDecayed": 0,
    "decayingModel": [85],
    "modelOverrides": {
        "threshold": 30
    }
    "score": 30,
}

```


Implementation in MISP: API query body
/attributes/restSearch



Creating a new decay algorithm (1)

The current architecture allows users to create their **own** formulae.

1. Create a new file `$filename` in `app/Model/DecayingModelsFormulas/`
2. Extend the Base class as defined in `DecayingModelBase`
3. Implement the two mandatory functions `computeScore` and `isDecayed` using your own formula/algorithm
4. Create a Model and set the formula field to `$filename`

Use cases:

- Add support for **more feature** (expiration taxonomy)
- **Query external services** then influence the score
- Completely **different approach** (i.e streaming algorithm)
- ...

Creating a new decay algorithm (1)
The current architecture allows users to create their **own** formulae.
1. Create a new file `$filename` in `app/Model/DecayingModelsFormulas/`
2. Extend the Base class as defined in `DecayingModelBase`
3. Implement the two mandatory functions `computeScore` and `isDecayed` using your own formula/algorithm
4. Create a Model and set the formula field to `$filename`
Use cases:
- Add support for **more feature** (expiration taxonomy)
- **Query external services** then influence the score
- Completely **different approach** (i.e streaming algorithm)
- ...

Creating a new decay algorithm (2)

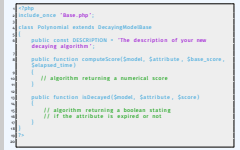
```php
<?php
include_once 'Base.php';

class Polynomial extends DecayingModelBase
{
    public const DESCRIPTION = 'The description of your new decaying algorithm';

    public function computeScore($model, $attribute, $base_score, $elapsed_time)
    {
       // algorithm returning a numerical score
    }

    public function isDecayed($model, $attribute, $score)
    {
        // algorithm returning a boolean stating
        // if the attribute is expired or not
    }
}
?>
```


Creating a new decay algorithm (2)



Decaying Models 2.0

- Improved support of *Sightings*
  - `False positive` *Sightings* should somehow reduce the score
  - `Expiration` *Sightings* should mark the attribute as decayed
- Potential *Model* improvements
  - Instead of resetting the score to `base_score` once a *Sighting* is set, the score should be increased additively (based on a defined coefficient); thus **prioritizing surges** rather than infrequent *Sightings*
  - Take into account related *Tags* or *Correlations* when computing score
- Increase *Taxonomy* coverage
  - Users should be able to manually override the `numerical_value` of *Tags*
- For specific type, take into account data from other services
  - Could fetch data from *BGP ranking*, *Virus Total*, *Passive X* for IP/domain/... and adapt the score

Decaying Models 2.0
- Improved support of Sightings
  - False positive Sightings should somehow reduce the score
  - Expiration Sightings should mark the attribute as decayed
- Potential Model improvements
  - Instead of resetting the score to base_score once a Sighting is set, the score should be increased additively (based on a defined coefficient); thus **prioritizing surges** rather than infrequent Sightings
  - Take into account related Tags or Correlations when computing score
- Increase Taxonomy coverage
  - Users should be able to manually override the numerical_value of Tags
- For specific type, take into account data from other services
  - Could fetch data from *BGP ranking*, *Virus Total*, *Passive X* for IP/domain/... and adapt the score