MISP and Decaying of Indicators

An indicator scoring method and ongoing imple-

Team CIRCL

info@circl.lu

April 15, 2024

MISP
Threat Sharing

Expiring IOCs: Why and How?

- **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

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

- 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)

  $\rightarrow$ Can be partially solved with *Taxonomies*

- 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

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**

ORGANISATIONS OPT-IN - SETTING A LEVEL OF CONFIDENCE
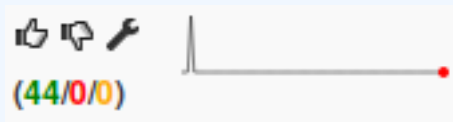
MISP is a peer-to-peer system, information passes through multiple instances.

- **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

$\rightarrow$ Achieved thanks to *Taxonomies*

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 (2)

→ Cherry-pick allowed *Tags*

■ 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 **?**
Deliberately deceptive	0 **?**

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

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

Where,

- $\text{score} \in [0, +\infty$
- $\text{base\_score} \in [0, 100]$
- 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

CURRENT IMPLEMENTATION IN MISP

■ Decay score toggle button

▶ Shows Score for each *Models* associated to the *Attribute* type

/attributes/restSearch

```
1  "Attribute": [
2    {
3      "category": "Network activity",
4      "type": "ip-src",
5      "to_ids": true,
6      "timestamp": "1565703507",
7      [...]
8      "value": "8.8.8.8",
9      "decay_score": [
10       {
11         "score": 54.475223849544456,
12         "decayed": false,
13         "DecayingModel": {
14           "id": "85",
15           "name": "NIDS Simple Decaying Model"
16         }
17       }
18     ],
19 [...]
```

- **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

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

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

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

Where,

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

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

Current implentation ignores `source_confidence`:

$$\rightarrow \texttt{base\_score} = \textit{tags}$$

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

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

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

$\rightarrow$ `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`

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

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

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

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

Create, modify, visualise, perform mapping

Simulate *Attributes* with different *Models*

/attributes/restSearch

```
1  {
2      "includeDecayScore": 1,
3      "includeFullModel": 0,
4      "excludeDecayed": 0,
5      "decayingModel": [85],
6      "modelOverrides": {
7          "threshold": 30
8      }
9      "score": 30,
10 }
11
```

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)
- ...

```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
    }
}
?>
```

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