MISP Object Template

Building custom and open data models

CIRCL / Team MISP Project

http://www.misp-project.org/
Twitter: @MISPProject

NSPA

MISP
Threat Sharing

Objects - or How We Learned to Stop Worrying and Love the Templates

- Attributes are a simple but powerful tool to describe data
- Lacking the capability to create containers around attributes describing a common concept
- The goal was to develop something semi-standardised, with the option to **dynamically build templates**
- We have considered a list of different solutions such as simple boolean operators, but found that the current implementation was superior.
- The result is a simple template that uses the basic attriubte types as building blocks along with some meta data
- The template does **not have to be known** in order to use the constructed objects
- What we maintain now is a set of common objects, but similarly to our other JSON formats, users can extend it with their own ideas.

MISP Object Templates

- Using a similar JSON format as the taxonomies, galaxies, warninglists.
- You can find the default set of object templates in the git repository[1].
- Some of the object templates capture objects from other standards or mimic the output of tools
- We tried to capture the most common use-cases coming from our own use-case as well as those of various partners that got involved
- Improvements or pull requests for new object templates are of course always welcome

[1] https://www.github.com/MISP/misp-objects/

Existing Object examples

- AIL-leak - **AIL object, an example for an object catering to the output of another tool**
- Android permission - **An object used to further contextualise another object**
- Bank account
- File **Generic object to describe a file**
- Passive DNS
- Regex
- Sandbox report
- Vulnerability **Enabling new use-cases such as pre-sharing of vulnerability information**
- x509
- Yara **Verbatim sharing of rule sets along with meta-data**

Object Template skeleton

```
{
  "requiredOneOf": [],
  "required": [],
  "attributes": {},
  "version": 1,
  "description": "My description",
  "meta-category": "Chosen meta category",
  "uuid": "Object template uuid",
  "name": "Object template name"
}
```

Adding elements to an object template

```
"regexp-type": {
  "description": "Type of the regular expression syntax.",
  "disable_correlation": true,
  "ui-priority": 0,
  "misp-attribute": "text",
  "values_list": [
    "PCRE",
    "PCRE2",
    "POSIX BRE",
    "POSIX ERE"
  ]
},
```

Attribute keys

- Primary key: Object relation
- description: A description of the attribute in relation to the object
- disable_correlation: You can disable correlations for attributes in the resulting object
- ui-priority: Not implemented yet, but the idea is to have a "quick view" of objects only showing certain prio levels
- misp-attribute: The misp attribute type used as as the building block
- values_list: an optional list of values from which the user **must** choose instead of entering a value manually
- sane_defaults: an optional list of values from which the user **may** choose instead of entering a value
- multiple: Allow the user to add **more** than one of this attribute

Enforcement of certain keys

- The template also defines which of the added attributes are mandatory
- Requirements are pointed to via their **object relations names**
- We differentiate between two types of rule sets:
  - Required: Everything in this list has to be set in order for the object to validate
  - Required One Of: Any of the attributes in this list will satisfy the requirements

What will the template actually do?

- Templates create a form that can be used to populate an event
- When using templates, MISP will enforce everything according to the template rules
- However, these are only optional, users can avoid using the templates when creating events via the API
- The reason for this is that you do not need to have the template in order to create an object
- The limitation of this system: You **cannot modify** objects that were created with unknown templates

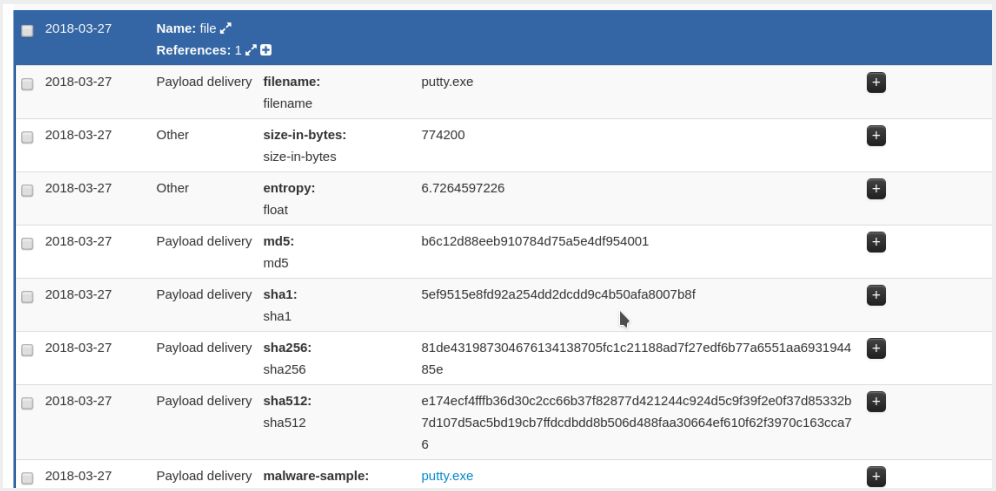
Templates as rendered in the UI

Add File Object

Object Template	File v10
Description	File object describing a file with meta-information
Requirements	**Required one of**: filename, size-in-bytes, authentihash, ssdeep, imphash, pehash, md5, sha1, sha224, sha256, sha384, sha512, sha512/224, sha512/256, tlsh, pattern-in-file, x509-fingerprint-sha1, malware-sample
Meta category	File
Distribution	Inherit event
Comment	

Save	Name :: type	Description	Category	Value
☐	**Md5** :: md5	[Insecure] MD5 hash (128 bits)	Payload delivery	
☐	**Pattern-in-file** :: pattern-in-file	Pattern that can be found in the file	Payload installation	
☐	**Sha256** :: sha256	Secure Hash Algorithm 2 (256 bits)	Payload delivery	
☐	**Sha512** :: sha512	Secure Hash Algorithm 2 (512 bits)	Payload delivery	
☐	**Filename** :: filename	Filename on		

Templates as rendered in the UI



2018-03-27	**Name:** file **References:** 1			
2018-03-27	Payload delivery	**filename:** filename	putty.exe	+
2018-03-27	Other	**size-in-bytes:** size-in-bytes	774200	+
2018-03-27	Other	**entropy:** float	6.7264597226	+
2018-03-27	Payload delivery	**md5:** md5	b6c12d88eeb910784d75a5e4df954001	+
2018-03-27	Payload delivery	**sha1:** sha1	5ef9515e8fd92a254dd2dcdd9c4b50afa8007b8f	+
2018-03-27	Payload delivery	**sha256:** sha256	81de43198730467613413870 5fc1c21188ad7f27edf6b77a6551aa693194485e	+
2018-03-27	Payload delivery	**sha512:** sha512	e174ecf4fffb36d30c2cc66b37f82877d421244c924d5c9f39f2e0f37d85332b7d107d5ac5bd19cb7ffdcdbdd8b506d488faa30664ef610f62f3970c163cca76	+
2018-03-27	Payload delivery	**malware-sample:**	putty.exe	+

Q&A

- https://github.com/MISP/MISP
- https://github.com/MISP/misp-objects
- info@circl.lu (if you want to join one of the MISP community operated by CIRCL)
- PGP key fingerprint: CA57 2205 C002 4E06 BA70 BE89 EAAD CFFC 22BD 4CD5