Proposal: Arbitrary property filtering

Marlon – DHS / Emmanuelle – MITRE

February 12th, 2019

Goals

- Expand filtering capabilities via matching against any object property.
- It would address:
  - The relationship pivoting use case
    - Examples: 1, 2*
  - The ability to filter upon specific information if an ID is not known in advance
    - Examples: 3*, 4, 6
  - Filtering on TLP Markings*
    - Example: 5
  - Filtering on confidence values*
    - Example: 6
  - Identify sighted indicator
    - Example: 7
  - The ability to query internal references
    - Example: 8

*This is a joint AIS and Federal community use-case

Goals (continued)

- **No** changes to where URL Parameters are available
- **No** changes to Supported Fields for Match (§ 3.4.1)
- This is **not** a substitution/solution to a full TAXII Query capability

Types of queries enabled by this proposal

1. ?match[type]=relationship&match[target_ref]=<identifier>
2. ?match[relationship_type]=mitigates&match[target_ref]=<identifier>*
3. ?match[external_id]=CVE-2016-1234*
4. ?match[type]=observed-data&match[value]=1.2.3.4
5. ?match[object_marking_refs]=marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da,marking-definition--f88d31f6-486f-44da-b317-01333bde0b82&match[type]=indicator&match[created_by_ref]=<identifier>*
6. ?match[type]=indicator&match[confidence]=90,91,92,93,94,95,96,97,98,99,100*
7. ?match[type]=sighting&match[sighting_of_ref]=<identifier>
8. ?match[type]=report&match[object_refs]=<identifier>

Example 1

- ?match[type]=relationship&match[target_ref]=<identifier>
- ?match[type]=relationship&match[source_ref]=<identifier>
- “Given an ID, I want to know what relationships point to it“ [2][3]
- “Given an ID, I want to know what relationships from to it“


```
{
  "type": "relationship",
  "spec_version": "2.1",
  "id": "relationship--df7c87eb-75d2-4948-af81-9d49d246f301",
  "created": "2016-04-06T20:06:37.000Z",
  "modified": "2016-04-06T20:06:37.000Z",
  "relationship_type": "indicates",
  "source_ref": "indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
  "target_ref": "malware--9c4638ec-f1de-4ddb-abf4-1b760417654e"
}
```

```
{
  "type": "relationship",
  "spec_version": "2.1",
  "id": "relationship--df7c87eb-75d2-4948-af81-9d49d246f301",
  "created": "2016-04-06T20:06:37.000Z",
  "modified": "2016-04-06T20:06:37.000Z",
  "relationship_type": "indicates",
  "source_ref": "indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
  "target_ref": "malware--9c4638ec-f1de-4ddb-abf4-1b760417654e"
}
```

Example 2

- ?match[relationship_type]=mitigates&match[target_ref]=<vulnerability--identifier>
- “Get all mitigation relationships that target a specific vulnerability” *


```
{
   "type": "relationship",
   "spec_version": "2.1",
   "id": "relationship--df7c87eb-75d2-4948-af81-9d49d246f301",
   "created": "2016-04-06T20:06:37.000Z",
   "modified": "2016-04-06T20:06:37.000Z",
   "relationship_type": "mitigates",
   "source_ref": "course-of-action--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
   "target_ref": "vulnerability--9c4638ec-f1de-4ddb-abf4-1b760417654e"
}
```

Example 3

- ?match[external_id]=CVE-2016-1234
- “Get intel on any object that references CVE-2016-1234” *


```
{
  "type": "vulnerability",
  "spec_version": "2.1",
  "id": "vulnerability--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created": "2016-05-12T08:17:27.000Z",
  "modified": "2016-05-12T08:17:27.000Z",
  "name": "CVE-2016-1234",
  "external_references": [
    {
      "source_name": "cve",
      "external_id": "CVE-2016-1234"
    }
  ]
}
```

Example 4

- ?match[type]=observed-data&match[value]=1.2.3.4

- "Does anyone have intel on IP 1.2.3.4?" [1][6]


```
{
  "type": "observed-data",
  "spec_version": "2.1",
  "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
  "created_by_ref": "identity--311b2d2d-f010-4473-83ec-1edf84858f4c",
  "created": "2016-04-06T19:58:16.000Z",
  "modified": "2016-04-06T19:58:16.000Z",
  "first_observed": "2015-12-21T19:00:00Z",
  "last_observed": "2015-12-21T19:00:00Z",
  "number_observed": 50,
  "objects": {
    "0": {
      "type": "ipv4-addr",
      "value": "1.2.3.4"
    }
  }
}
```

Example 5

- ?match[object_marking_refs]=marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9,marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da&match[type]=indicator&match[created_by_ref]=<identity--identifier>
- “Get all TLP:WHITE OR TLP:GREEN indicators created by a specific entity” *


```
{
  "type": "indicator",
  "spec_version": "2.1",
  "id": "indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
  "created_by_ref": "identity--311b2d2d-f010-4473-83ec-1edf84858f4c",
  "created": "2017-01-01T00:00:01.000Z",
  "modified": "2017-01-01T00:00:01.000Z",
  "indicator_types": [
    "malicious-activity"
  ],
  "pattern": "[file:hashes.MD5 = 'd41d8cd98f00b204e9800998ecf8427e']",
  "valid_from": "1970-01-01T00:00:01Z",
  "object_marking_refs": [
    "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9"
  ]
}
```

Example 6

- ?match[type]=indicator&match[confidence]=90,91,92,93,94,95,96,97,98,99,100
- “Get all indicators with high confidence” *


```
{
  "type": "indicator",
  "spec_version": "2.1",
  "id": "indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
  "created": "2017-01-01T00:00:01.000Z",
  "modified": "2017-01-01T00:00:01.000Z",
  "indicator_types": [
    "malicious-activity"
  ],
  "pattern": "[file:hashes.MD5 = 'd41d8cd98f00b204e9800998ecf8427e']",
  "valid_from": "1970-01-01T00:00:01Z",
  "confidence": 91
}
```

Example 7

- ?match[type]=sighting&match[sighting_of_ref]=<indicator--identifier>
- “I have an indicator and I want to pull all sightings.” [4]


```
{
   "type": "sighting",
   "spec_version": "2.1",
   "id": "sighting--bfbc19db-ec35-4e45-beed-f8bde2a772fb",
   "created": "2016-04-06T20:06:37.000Z",
   "modified": "2016-04-06T20:06:37.000Z",
   "sighting_of_ref": "indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
   "where_sighted_refs": [
      "identity--311b2d2d-f010-4473-83ec-1edf84858f4c"
   ]
}
```

Example 8

- ?match[type]=report&match[object_refs]=<identifier>
- “Get any report object that contains a link to this SDO/SRO” [5]


```
{
  "type": "report",
  "spec_version": "2.1",
  "id": "report--84e4d88f-44ea-4bcd-bbf3-b2c1c320bcb3",
  "created_by_ref": "identity--311b2d2d-f010-4473-83ec-1edf84858f4c",
  "created": "2015-12-21T19:59:11.000Z",
  "modified": "2015-12-21T19:59:11.000Z",
  "name": "The Black Vine Cyberespionage Group",
  "description": "A simple report with an indicator and campaign",
  "report_types": [
    "campaign"
  ],
  "published": "2016-01-20T17:00:00Z",
  "object_refs": [
    "indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
    "campaign--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "relationship--df7c87eb-75d2-4948-af81-9d49d246f301"
  ]
}
```

What would need to be changed? (§ 3.4)

`match[<field>]`	The `match` parameter defines filtering on the specified `<field>`. The list of fields that **MUST** be supported is defined per Endpoint as defined in sections 4, 5, and 6. The `match` parameter can be specified any number of times, where each `match` instance specifies an additional filter to be applied to the resulting data and each `<field>` **MUST NOT** occur more than once in a request. Said another way, all match fields are ANDed together.
	
	All `<field>` parameters are defined in the following table. Requests **MAY** use a `<field>` not defined in this specification, and servers **MAY** ignore fields they do not understand.
	
	Each field **MAY** contain one or more values. Multiple values are separated by a comma (U+002C COMMA, ",") without any spaces. If multiple values are present, the match is treated as a logical OR. For instance, `?match[type]=incident,malware` specifies a filter for objects that are of type incident OR malware.
	
	Examples
	`?match[type]=campaign,malware,threat-actor`
	`?match[type]=incident&match[version]=2016-01-01T01:01:01.000Z`

What would need to be changed? (Cont.)

Any <field> parameter not defined under Section 3.4.1 **MUST** be treated as any regular property check on an object. If an object does not include the property it is simply ignored (not included) as part of the final result set.

Algorithm (Part 1)

```
function filter_property(collection_objs, property, value):
    result_set <- {}
    for each object in collection_objs:
        if recursive_search(object, property, value) is True:
            result_set.add(object)
    return result_set
```

Algorithm (Part 2)

```
function recursive_search(object, property, value):
    if property in object and object[property] == value:
        return True
    for each object_key, object_value in object:
        if object_value is dict:
            if recursive_search(object_value, property, value) is True:
                    return True
        else if object_value is list:
            for each element in object_value:
                if element is dict:
                    if recursive_search(element, property, value) is True:
                        return True
                else if element == value:
                    return True
    return False
```

Reference to GitHub issues/Slack

1. (G #4) RFE: TAXII Observed Data Query. https://github.com/oasis-tcs/cti-taxii2/issues/4
2. (G #6) Add ability to find all objects related to a particular STIX object ID, to prevent an indeterminate number of queries to find them all. https://github.com/oasis-tcs/cti-taxii2/issues/6
3. (G #7) Need ability to request related objects in one request to a distance of 1(?). https://github.com/oasis-tcs/cti-taxii2/issues/7
4. (G #15) As a User, I want to traverse the STIX graph over TAXII in an efficient manner, so I don't waste resources. https://github.com/oasis-tcs/cti-taxii2/issues/15

Reference to GitHub issues/Slack

5. (G #68) No way to query internal references. https://github.com/oasis-tcs/cti-taxii2/issues/68
6. Slack Channel conversation on #taxii. https://cti-tc.slack.com/messages/C0DS7CXV5/