STIX 2.1 Specification

Working Concepts

Document Table of Contents

Document Development Status

1. Draft: The development work on this topic is complete and it has been reviewed and discussed on a Full TC Call. The topic is now in the final documents.
2. Review: The development work is done and the topic needs to be reviewed and a it needs to be proposed to the Full TC to add to the final documents.
3. Development: The development work for this topic is underway and work is being done during working calls, email, and slack.
4. Mini-Group: A mini-group for this topic is formed and work is being done during in that group.
5. Planned: The topic is accepted into the release but has not yet been officially picked up by the TC to be worked on. The topic may have some concepts or proposals, but it is not an official work product yet.
6. Paused: The community has discussed this item, but has not created a mini-group or consensus proposal. Paused work was started but is on hold to focus on other topics.

Concepts	Status
`confidence`	Draft
`internationalization`	Review
`location`	Development
Objects	**Status**
`asset`	Paused
`event`	Planned

infrastructure	Mini-Group
malware-instance	Development
intel-note	Review
opinion	Review

1. Cross-Cutting Changes

1.1. Confidence

Appendix XXX

This table provides the mappings that **MUST** be used when translating to and from the 0-100 scale to other scales for confidence.

None/ Low/ Med/ High	STIX Confidence Value	Range of Values
Not Specified	Not Specified	
None	0	0
Low	15	1-29
Med	50	30-69
High	85	70-100

0-10 Scale	STIX Confidence Value	Range of Values
Not Specified	Not Specified	
0	0	0-4
1	10	5-14
2	20	15-24
3	30	25-34
4	40	35-44

5	50	45-54
6	60	55-64
7	70	65-74
8	80	75-84
9	90	85-94
10	100	95-100

Admiralty Credibility	STIX Confidence Value	Range of Values
6 - Truth cannot be judged	(Not present)	N/A
5 - Improbable	10	0-19
4 - Doubtful	30	20-39
3 - Possibly True	50	40-59
2 - Probably True	70	60-79
1 - Confirmed by other sources	90	80-100

WEP	STIX Confidence Value	Range of Values
Impossible	0	0
Highly Unlikely/Almost Certainly Not	10	1-19
Unlikely/Probably Not	30	20-39
Even Chance	50	40-59
Likely/Probable	70	60-79
Highly likely/Almost Certain	90	80-99
Certain	100	100

https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/books-and-monographs/sherman-kent-and-the-board-of-national-estimates-collected-essays/6words.html
https://en.wikipedia.org/wiki/Words_of_estimative_probability

DNI Scale	STIX Confidence Value	Range of Values

Almost No Chance / Remote	5	0-9
Very Unlikely / Highly Improbable	15	10-19
Unlikely / Improbable	30	20-39
Roughly Even Change / Roughly Even Odds	50	40-59
Likely / Probable	70	60-79
Very Likely / Highly Probable	85	80-89
Almost Certain / Nearly Certain	95	90-100

NOTES:

Confidence - The trust in the data behind the intelligence / the accuracy of the intelligence

Credibility - The trust in the source providing the intelligence

Severity - The criticality level of the exploit / malware / incident / event

Relevance - How relevant the exploit / malware / incident / event is to your organization

Likelihood probability (ICD 203) is also regularly used.
https://github.com/MISP/misp-taxonomies/blob/master/estimative-language/machinetag.json

"Business Impact" is both a (human) Mitigation Analyst and machine driven/enhanced determination.

NOTES from Pat:
For example, Company "A" and "B" receive CTI stating 1.2.3.4 is "bad". Company "A's" CTI automated processes check on ingestion and find that they have not seen any activity to/from 1.2.3.4 in the last 12 months. A "Business Impact" rating of ("0","None",".001", "Low", pick your scale ;-) can be assigned and used in the decision process for automated blocking. In this case a Mitigation Analyst review is not required. Company "B's" CTI automated processes check on ingestion and find significant ongoing activity. A high value for potential "Business Impact" is calculated. These triggers the workflow to send to a Mitigation Analyst for review. The other received subjective measures (High Certainty, High Confidence) along with high rating for this Source could influence these automated ingestion decisions and both block the activity and send to to the Operational Mitigation Work-flow.

The mitigation analyst could discover that "1.2.3.4" belongs to one it's prime web services customers and that blocking it would severely impact business revenue/relationships.

In another "real world" example, it has been determined by very good analysts that actor "X", ALWAYS uses highly randomized Google.Com email addresses to target sector "Y", AND there's a high risk active zero day attack underway. Company "A" disallows use of Google.com email address for business purposes (sender or receiver) and sets "Business Impact" accordingly. Company "B" solely uses Google for all of it's business communication and collaboration.

Comment from Allan:

The challenge with this table is that if you dont keep the original value as well as the mapping value then you will not be able to map between the columns without errors.

Vendor A creates Admiralty Score 1
Vendor A sends 100
Vendor B receives 100 and then maps to Low/Med/High scale
Vendor B sends to Vendor C High
Vendor C creates intel score of 67 based on rating of High starts at that low number
Vendor C sends intel 67 back to Vendor A who maps that to Admiralty score 2 (not 1).

Now this problem does not exist if everyone shares the original STIX score 0-100 **and** their mapping score/value together.

However, the problems start when they don't. As soon as you map between one numerical system and another score system the problems will be encountered.
and given that it will be based on an indirect communication path as I describe above it will be hard to detect the errors. Therefore, to avoid these problems I suggest the following consideration:

a) require the number 0-100 score
b) allow optional key/value pair for the optional mappings provided by the vendor

score: 60
mapping: [{ as-score, 2 }, { wep-score, 'likely' }]

and then effectively both are provided and avoids the chain problem I see above.

1.2. Location

Open Questions:

1. Is location a separate SDO or properties on existing SDOs (identity, intrusion-set, potentially others)?
    a. If a separate SDO, do you link directly to it or do you always link via Identity?
2. Is all of the data (including civic addresses) represented within the GeoJSON content or do we separate out GeoJSON from the rest?
3. How much detail do we include?

Type Name: `location`

Location is used to describe geographic locations. It supports describing by general region, civic address, or using GeoJSON.

At least one of the **`region`**, **`address`**, and **`geojson`** properties **MUST** be present.

1.2.1. Properties

Location Specific Properties
`region, address, geojson`

Property Name	Type	Description
region (optional)	`open-vocab`	The region that this location describes. This property **SHOULD** contain a value from `region-ov`.
address (optional)	`address`	The address of this location.
geojson (optional)	`geojson`	A GeoJSON description of this location. This object **MUST** conform to the GeoJSON specification per RFC7946.

Type Name: `address`

Address is a sub-type used only by location and is used to describe civic (street) addresses.

Address Specific Properties		
country, administrative_area, city, address, postal_code		
Property Name	**Type**	**Description**
country (required)	`string`	The country that this location describes. This property **MUST** contain a valid ISO ALPHA-2 Code.
administrative_area (optional)	`string`	The state, province, or other sub-national administrative area that this location describes. This field **SHOULD NOT** contain abbreviated forms of administrative areas (e.g. use "New York", not "NY").
city (optional)	`string`	The city that this location describes. This field **SHOULD NOT** contain abbreviated forms of city names (e.g. use "New York City", not "NYC").
address (optional)	`string`	All lines of the street address that this location describes. Line breaks are permitted.

postal_code (optional)	`string`	The postal code that this location describes.

1.2.2. Modifications to Other Objects

1.2.2.1. Identity

location (optional)	`location`	The geographic location of this identity.

1.2.2.2. Malware

Add relationship "authored_by" to a threat actor. Can also use "used_by".

1.2.2.3. Intrusion Set

locations (optional)	`list` of type `location`	The geographic locations of this intrusion set.

1.2.2.4. Threat Actor, Sighting

No modifications necessary, existing relationships to identities already capture this information.

2. Potential Domain Objects

2.1. Event

Type Name: `event`

An incident is a violation of an explicit or implied security policy [TODO add ref to NIST]. Prior to a confirmed incident, many organizations call incidents "events" or "investigations". Investigations can include, but are not limited to:

- attempts (either failed or successful) to gain unauthorized access to a system or its data
- unwanted disruption or denial of service
- the unauthorized use of a system for the processing or storage of data
- changes to system hardware, firmware, or software characteristics without the owner's knowledge, instruction, or consent

For example, an Investigation could describe a malware infestation on one of a company's laptops.

The Event SDO represents data both about full incidents as well as unconfirmed incidents and investigations. It might represent an investigation or incident at a single

Sentence from Observed Data: Observed Data can also be related to Incident to provide data that was discovered during the response process or that is otherwise part of the incident.

Open Questions:

1. What are the use cases that we need to solve?
    a. Something that happens in an organization that potentially has security impact (event investigation).
    b. Something that happens in an organization that actually has security impact (incident investigation).
    c. Something that happens outside an organization (external event investigation)
    d. A package of early-stage analysis, or an evolving analysis, like a MISP event (or CRITS event).
2. What is the best way to solve use case (d) - MISP event?
    a. Could we improve the report type to support the event properties? What would need to change in report to make this happen?
    b. If a separate Event object is used to communicate a security analysis report, when and why would I use an Event SDO instead of a Report SDO?
    c. One use case MISP has related to honeypot data sounds like it overlaps with Jason Keirstead's notional Classification SDO, i.e., "Here's a blob of observable data but not linked to an indicator sighting."
3. What is the best case to solve use case (a, b, c) - investigation?
    a. New object?
    b. How do we represent the different timestamps and contacts, which can differ by org/product?
    c. How do we relate this incident SDO to the Report SDO and/or observed data?
    d. How and what level of detail do you capture impact information (related to targeting)?
4. What should we call it? Incident? Investigation? Event?
5. What capabilities are important for STIX 2.1 and what can we add on later?
6. How do we capture risk level? Can we re-use whatever approach we use for indicator?
7. How do we capture impacts?
    a. Impacts can be to cyber assets, business operations, financials, etc.
    b. Impacts can be actual (a list of assets) or summarized (a number of assets).
    c. Do we need an asset SDO (friendly infrastructure)?
    d. Do we need to capture both counts and actual assets?
8. How do we capture victims?
9. Do you represent relationships to COAs as embedded relationships or separate relationship objects?

2.1.1. Properties

Common Properties		
TODO		
Incident Specific Properties		
TODO		
Property Name	**Type**	**Description**

type (required)	`string`	The value of this field **MUST** be `investigation`
labels (required)	`list` of type `open-vocab`	This property is a list of classifications for the Incident. This is an open vocabulary and values **SHOULD** come from the `investigation-label-ov` vocabulary.
name (required)	`string`	A name used to identify the Event.
description (optional)	`string`	A description that provides more details and context about the Event, potentially including its purpose and its key characteristics.
aliases (optional)	`list` of type `string`	Additional names or titles for this incident.
status (required)	`open-vocab`	open, closed, triaged, contained, mitigated, remediated, in_remediation, dismissed
timestamps	`dictionary`	Timestamps represent timestamps relevant to the lifecycle of this investigation. The keys of this dictionary **SHOULD** be values from the `event-timestamps-ov` open vocabulary. The values of this dictionary **MUST** be `timestamp`s representing the data. Current vocab values: reported, triaged, started, detected, contained, remediated
contacts	`dictionary`	Contacts represent points of contact for this investigation. They keys of this dictionary **SHOULD** be values from the `event-contacts-ov` open vocabulary. The values of this dictionary **MUST each** be a `list` of type `identifier`, where the identifiers are references to `identity` SDOs. Current vocab values: reported_by, detected_by, responder, coordinated_by, impacted_users, related_parties.
detected_with (optional)	`list` of type `open-vocab`	How was this event detected.
number_systems (optional)	`integer`	The number of affected systems
number_user (optional)	`integer`	The number of affected users

number_records (optional)	integer	The number of affected records
activity (optional)	list of type event-activity	Captures a journal of activity that has been taken in the course of this event investigation.

2.1.2. Event Activity Type

Property Name	Type	Description
activity_date	timestamp	
activity	string	

2.1.3. Relationships

These are the relationships explicitly defined between the Event object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Incident object by way of the Relationship Object. The reverse relationships (relationships "to" the Incident object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the related-to relationship name or, as with open vocabularies, user-defined names.

Embedded Relationships			
created_by_ref	source		
object_markings_refs	marking-definition		
Common Relationships			
duplicate-of, derived-from, related-to			
Source	**Name**	**Target**	**Description**
event	attributed-to	campaign, intrusion-set, threat-actor	This Relationship describes that the the related Campaign, Intrusion Set, or Threat Actor is responsible for the Incident. For example, an attributed-to Relationship from an Incident to a Campaign means that the Campaign was used to carry out the incident.
Reverse Relationships			

observed-data	part-of	event	See forward relationship for definition.
course-of-action	contains	event	See forward relationship for definition.
course-of-action	mitigates	event	See forward relationship for definition.
observed-data	part-of	event	This Relationship documents that this Observed Data is a part of the related Incident. For example, a part-of Relationship linking a set of Observed Data containing network connection information to an Incident could capture network traffic that originated from a compromised host and was determined to be command and control traffic.
identity	victim-of	event	This Relationship describes that the the related Victim Target was a victim of this incident. For example, an part-of Relationship from an Incident to a Victim Target representing ACME Corporation means that ACME Corporation was an actual victim of that Incident.
malware, attack-pattern, tool, vulnerability	used-in	event	

2.1.4. Examples

```
{
  "type": "investigation",
  "id": "investigation--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
  "created_by_ref": "source--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T20:03:48.000Z",
  "modified": "2016-04-06T20:03:48.000Z",
  "labels": "investigation",
  "name": "Green Group Infiltration of Web Servers",
  "description": "Green group was able to infiltrate the web server infrastructure and caused
sporadic and unpredictable content defacement issues."
  "timestamps": {
    "reported": "2016-04-06T20:03:48.000Z",
```

```
  },
  "status": "opened"
}
```

2.2. Infrastructure

Type Name: `infrastructure`

Malicious infrastructures are a type of TTP that describes the backend services and resources used by attackers to carry out attacks. Command and control servers, malware delivery sites, and phishing sites are examples of malicious infrastructure.

The Infrastructure SDO contains basic descriptive information and a characterization of the technical details of the infrastructure using Cyber Observables. Relationships to and from Malicious Infrastructure can relate it to the attackers (Threat Actors, Intrusion Sets, and Campaigns) and incidents that use it and Indicators that can detect it.

The Infrastructure SDO **MUST NOT** be used to capture information about defender infrastructures or assets.

2.2.1. Use Cases

1. An infrastructure can have systems or IP addresses come and go all the time. But it is a single infrastructure.
2. Malicious infrastructures can be setup just in time or can be used across multiple Campaigns or used by multiple Threat Actors.
3. Malware and Tools will use Infrastructures for command and control, data exfiltration, delivery, etc.
4. An infrastructure can include external or internal systems as they are used by the Threat Actor.
5. Characterizing malicious infrastructure used by adversaries
    a. C2
        i. Malicious web hosting/compromised web servers
            1. Domain name re-use by multiple threat actors/campaigns
            2. Domain names registered by the same threat actors (e.g., DNS SOA records)
            3. Domain names with common whois data points.
    b. Botnets
        i. Malware hosting
        ii. Specific botnet components
            1. Target-list hosting domain/IP(s)
            2. C2 domain/IP(s)
    c. Exploit kits/hosting
        i. Compromised domain --> hosts --> Malware (exploit kit)
        Malware (exploit kit) --> installs --> Malware (other malware)
    d. Malicious use of existing web services
        i. E.g., Twitter
    e. Digital signatures
        i. E.g., the set of digital signatures used to sign malware binaries

Notes about using Observed Data with things like Infrastructure or Malware.

1. The Infrastructure or Malware object will have Cyber Observable properties directly on them. These fields will allow you to capture the data that characterises these objects.
2. So say that an Infrastructure is known to exist in S.Korea and it is using LInux based Web Cameras as a delivery point for C-n-C. These IP addresses and the Make/Model of the Web Cams would all be on the Infrastructure Object itself.
3. You may need to revision the Infrastructure object multiple times as you find or discover more things. In this case, some fields on the Infrastructure object may need to be an array to allow for say thousands of IP address.
4. The way Observed Data fits in, is when you do a Sighting. When you want to say you saw an instance of these things.
5. You may not want to capture all of the technical details on the object if you feel they're too transitory. i.e. if a C2 network has a dynamic domain generation algorithm, capturing all of the actual domains it uses is probably not useful. You would instead (probably in text for now) just capture the algorithm itself

2.2.2. Open Questions

- What types of adversary infrastructure are essential for us to characterize for 2.1?
    - C2
        - HTTP
        - DNS
        - Twitter
        - IRC
        - etc
    - Delivery
        - Hosting
    - Exfil
        - HTTP
        - DNS
        - Twitter
        - IRC
        - etc
    - Email delivery
    - Watering hole
    - Compromised benign site
    - Darkweb markets ??? (Is this just malware infrastructure or is it adversary infrastructure?)
    - Digital signatures ???
- Which Cyber Observable Objects do we need to effectively characterize that infrastructure?
    - Network-related objects?
    - Host-related objects?
- Do we explicitly want to limit this to just malicious infrastructure via normative text and name or do we want to keep it open but just focus on the malicious infrastructure aspects for now?
    - In other words, do we foresee having two separate objects, one for attacker infrastructure, and one for benign infrastructure, or do we see addressing attacker infrastructure use cases now and augmenting the single infrastructure object for 2.2+ to address the benign infrastructure use cases?
- Is infrastructure some collection of data that evolves over time or are individual IPs and URLs captured as individual infrastructure objects?
    - What is a good way to resolve the "developer vs. analyst" debate on this? Is there some kind of novel approach that works for both?
- Should we try to describe volatile adversary infrastructure such as Domain Generation Algorithms?

    - This could potentially be expressed using a STIX Pattern
    - Is this a priority for 2.1?
- How do we ensure that indicators can be created to point to infrastructure without just duplicating data? If the infrastructure has technical characterizations of observable aspects of the data then it can be very duplicative.
- How detailed should we try to characterize components of malicious infrastructure?
    - E.g., should we differentiate between proxies vs. the network resources that live behind them?
- Should we try to characterize communication to/from malicious infrastructure?
    - E.g., bilateral malware C2 communications
- Does it need a name field? Do people name infrastructure?
- Do we need different types of malware<->infrastructure relationships?

TODO: need to capture location once we resolve that discussion

2.2.3. Properties

Common Properties		
TODO		
Infrastructure Specific Properties		
name, **description**, **kill_chain_phases**, **first_seen**		
Property Name	**Type**	**Description**
type (required)	`string`	The value of this field **MUST** be `infrastructure`
labels (required)	`list` of type `open-vocab`	The type of infrastructure being described. This is an open vocabulary and values **SHOULD** come from the `infrastructure-type-ov` vocabulary.
description (optional)	`string`	A description that provides more details and context about the malicious Infrastructure, potentially including its purpose and its key characteristics.
kill_chain_phases (optional)	`list` of type `kill-chain-phase`	The list of Kill Chain phases for which this Infrastructure is used.
first_seen (optional)	`timestamp`	The time that this malicious Infrastructure was first seen.

observable_details (required)	observable-objects	Specifies any data observed about the infrastructure, in terms of Cyber Observable Objects. For example, an IP range or domain name.

2.2.4. Infrastructure Type Vocabulary

Type Name: infrastructure-type-ov

A non-exhaustive enumeration of adversary infrastructure types.

Vocabulary Value	Description
compromised-domain	Specifies a domain that was compromised by an adversary and used for malicious hosting of some kind.
command-and-control	Specifies infrastructure used for command and control (C2). This is typically a domain name or IP address.
target-list-hosting	Specifies infrastructure used for hosting a list of targets for DDOS attacks, phishing, and other malicious activities. This is typically a domain name or IP address.
botnet	Specifies the membership/makeup of a botnet, in terms of the network addresses of the hosts that comprise the botnet.
exfiltration	Specifies infrastructure used as an endpoint for data exfiltration.
staging	Specifies infrastructure used for staging.
anonymization	Specific infrastructure used for anonymization, such as a proxy.

2.2.5. Relationships

These are the relationships explicitly defined between the Infrastructure object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Infrastructure object by way of the Relationship Object. The reverse relationships (relationships "to" the Infrastructure object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the related-to relationship name or, as with open vocabularies, user-defined names.

Embedded Relationships	
created_by_ref	source

object_markings_refs	marking-definition

Common Relationships			
duplicate-of, derived-from, related-to			
Source	**Name**	**Target**	**Description**
infrastructure	targets	victim-target, vulnerability	This Relationship documents that this malicious Infrastructure is being used to target this Victim Target or Vulnerability. For example, a targets Relationship linking an Infrastructure for a phishing hosting site to a Victim Target representing the retail sector indicates that the phishing hosting site is targeted at the retail sector.
infrastructure	supports	malware	The infrastructure is used to host a malware family or particular malware instance.
infrastructure	supports	infrastructure	The infrastructure is a component of some broader/overarching infrastructure.
Reverse Relationships			
indicator	indicates	infrastructure	See forward relationship for definition.
course-of-action	mitigates	infrastructure	See forward relationship for definition.
campaign, intrusion-set, malware, threat-actor, tool	uses	infrastructure	See forward relationship for definition.
tool	uses	infrastructure	This Relationship documents that this Tool uses the related infrastructure to perform its functions. For example, a uses Relationship linking a remote access Tool to an Infrastructure representing a proxy indicates that Tool is or can be used through that proxy.

observed-data	part-of	infrastructure	

2.2.7. Examples

Malware C2 Infrastructure

```
{
  "type":"infrastructure",
  "id":"infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d",
  "created":"2016-05-07T11:22:30.000000Z",
  "modified":"2016-05-07T11:22:30.000000Z",
  "labels":["command-and-control"],
  "observable_details":{
    "0":{
      "type":"ipv4-addr",
      "value":"198.51.100.2"
    }
  }
}

{
  "type": "relationship",
  "id": "relationship--7aebe2f0-28d6-48a2-9c3e-b0aaa60266ed",
  "created": "2016-05-09T08:17:27.000000Z",
  "modified": "2016-05-09T08:17:27.000000Z",
  "relationship_type": "used-by",
  "source_ref": "infrastructure--38c47d93-d984-4fd9-b87b-d69d0841628d"
  "target_ref": "malware--16f4f3f9-1b68-4abb-bb66-7639d49f1e30"
}

{
  "type": "malware",
  "id": "malware--16f4f3f9-1b68-4abb-bb66-7639d49f1e30",
  "created": "2016-05-08T14:31:09.000000Z",
  "modified": "2016-05-08T14:31:09.000000Z",
  "is_family": true,
  "labels":[
    "rat"
  ],
  "name": "Poison Ivy"
}
```

Malware & Target List Hosting Domain

```
{
  "type":"infrastructure",
  "id":"infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73",
  "created":"2016-11-22T09:22:30.000000Z",
  "modified":"2016-11-22T09:22:30.000000Z",
  "labels":["target-list-hosting"],
  "observable_details":{
    "0":{
      "type":"domain-name",
      "value":"example.com"
    }
  }
}
```

```
{
  "type": "relationship",
  "id": "relationship--37ac0c8d-f86d-4e56-aee9-914343959a4c",
  "created": "2016-11-23T08:17:27.000000Z",
  "modified": "2016-11-23T08:17:27.000000Z",
  "relationship_type": "used-by",
  "source_ref": "infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73"
  "target_ref": "malware--3a41e552-999b-4ad3-bedc-332b6d9ff80c"
}

{
  "type": "malware",
  "id": "malware--3a41e552-999b-4ad3-bedc-332b6d9ff80c",
  "created": "2016-11-12T14:31:09.000000Z",
  "modified": "2016-11-12T14:31:09.000000Z",
  "is_family": true,
  "name": "IMDDOS"
}
```


Malware Botnet Infrastructure


```
{
  "type":"infrastructure",
  "id":"infrastructure--78cc7b4b-c6ab-40d1-82eb-95a3059641da",
  "created":"2017-03-15T04:22:30.000000Z",
  "modified":"2017-03-15T04:22:30.000000Z",
  "labels":["botnet"],
  "observable_details":{
    "0":{
      "type":"ipv4-addr",
      "value":"198.51.100.2"
    },
    "1":{
      "type":"ipv4-addr",
      "value":"198.51.100.4"
    },
    "2":{
      "type":"ipv4-addr",
      "value":"198.51.100.7"
    }
  }
}

{
  "type": "relationship",
  "id": "relationship--edce6fe8-2ac7-49d6-bd57-3973a4f819b8",
  "created": "2017-03-16T22:17:27.000000Z",
  "modified": "2017-03-16T22:17:27.000000Z",
  "relationship_type": "part-of",
  "source_ref": "infrastructure--78cc7b4b-c6ab-40d1-82eb-95a3059641da"
  "target_ref": "malware--496cac0a-77ea-4da0-b913-88e553483c8d"
}

{
  "type": "malware",
  "id": "malware--496cac0a-77ea-4da0-b913-88e553483c8d",
  "created": "2017-03-10T07:31:09.000000Z",
  "modified": "2017-03-10T07:31:09.000000Z",
  "is_family": true,
  "labels":[
    "bot"
  ],
```

```
  "name": "Asprox"
}
```


Related/Component Botnet Infrastructure


```
{
  "type":"infrastructure",
  "id":"infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73",
  "created":"2016-11-22T09:22:30.000000Z",
  "modified":"2016-11-22T09:22:30.000000Z",
  "labels":["target-list-hosting"],
  "observable_details":{
    "0":{
      "type":"domain-name",
      "value":"example.com"
    }
  }
}

{
  "type":"infrastructure",
  "id":"infrastructure--e4ed271e-e023-45db-99e6-1f912e79bd06",
  "created":"2016-11-22T11:04:18.000000Z",
  "modified":"2016-11-22T11:04:18.000000Z",
  "labels":["command-and-control"],
  "observable_details":{
    "0":{
      "type":"domain-name",
      "value":"control.example.com"
    }
  }
}

{
  "type":"infrastructure",
  "id":"infrastructure--a3536537-456a-47b5-84dc-fb7c340959e8",
  "created":"2016-11-18T04:22:30.000000Z",
  "modified":"2016-11-18T04:22:30.000000Z",
  "labels":["botnet"],
  "observable_details":{
    "0":{
      "type":"ipv4-addr",
      "value":"198.51.100.3"
    },
    "1":{
      "type":"ipv4-addr",
      "value":"198.51.100.9"
    }
  }
}

{
  "type": "relationship",
  "id": "relationship--43f753d8-61e2-472e-918e-d7c58e2463e7",
  "created": "2016-11-25T13:37:27.000000Z",
  "modified": "2017-11-25T13:37:27.000000Z",
  "relationship_type": "component-of",
  "source_ref": "infrastructure--d09c50cf-5bab-465e-9e2d-543912148b73"
  "target_ref": "infrastructure--a3536537-456a-47b5-84dc-fb7c340959e8"
}

{
```

```
  "type": "relationship",
  "id": "relationship--8386f241-b583-4c59-9056-a3b0db596d93",
  "created": "2016-11-25T13:37:27.000000Z",
  "modified": "2017-11-25T13:37:27.000000Z",
  "relationship_type": "component-of",
  "source_ref": "infrastructure--e4ed271e-e023-45db-99e6-1f912e79bd06"
  "target_ref": "infrastructure--a3536537-456a-47b5-84dc-fb7c340959e8"
}
```

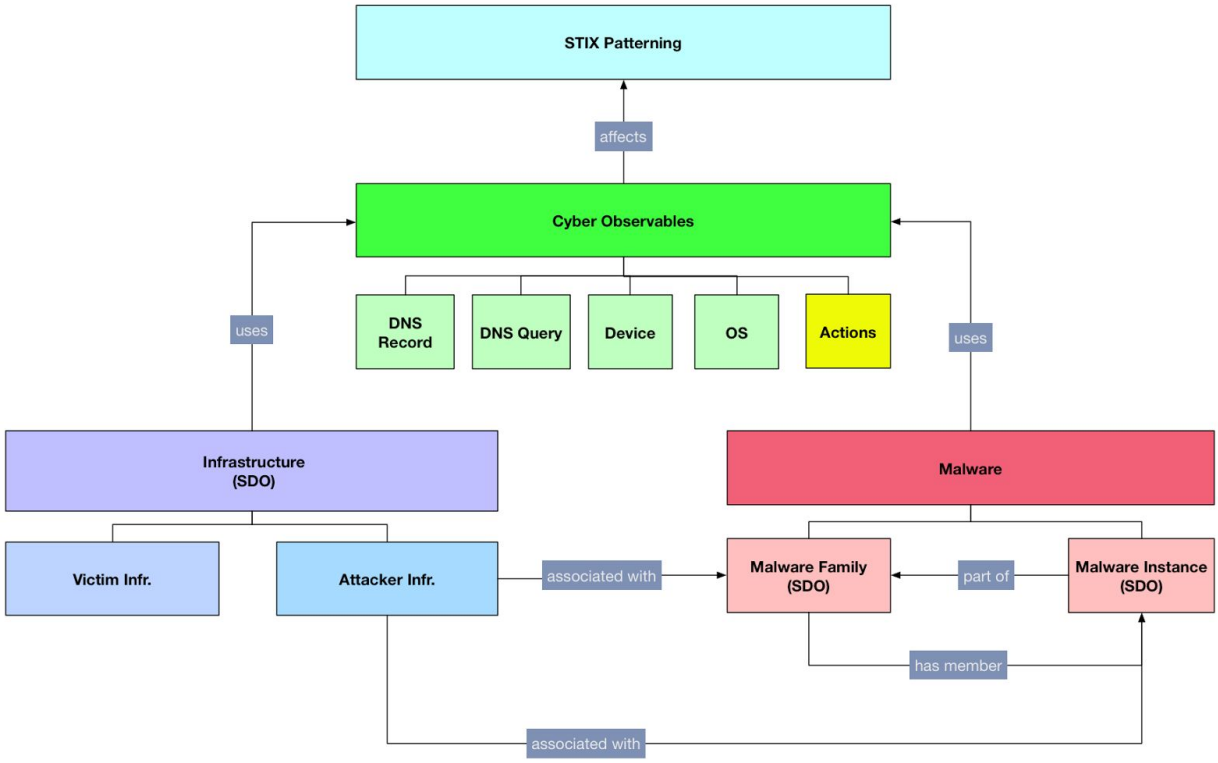
Malware Instance Hosted on Compromised Domain

```
{
  "type":"infrastructure",
  "id":"infrastructure--33588e0e-2bab-430e-9073-cacf704ea1e7",
  "created":"2017-04-04T13:01:21.000000Z",
  "modified":"2017-04-04T13:01:21.000000Z",
  "labels":["compromised-domain"],
  "observable_details":{
    "0":{
      "type":"domain-name",
      "value":"foo.example.com"
    }
  }
}

{
  "type": "relationship",
  "id": "relationship--8386f241-b583-4c59-9056-a3b0db596d93",
  "created": "2017-04-05T13:37:27.000000Z",
  "modified": "2017-04-05T13:37:27.000000Z",
  "relationship_type": "hosts",
  "source_ref": "infrastructure--33588e0e-2bab-430e-9073-cacf704ea1e7"
  "target_ref": "malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061"
}

{
  "type":"malware",
  "id":"malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created":"2016-05-12T08:17:27.000000Z",
  "modified":"2016-05-12T08:17:27.000000Z",
  "name":"SpyEye",
  "is_family":false,
  "labels":[
    "trojan"
  ],
  "sample_metadata":{
    "0":{
      "type":"file",
      "name":"cleansweep.exe",
      "size":126464,
      "hashes":{
        "MD5":"84714c100d2dfc88629531f6456b8276",
        "SHA-256":"861aa9c5ddcb5284e1ba4e5d7ebacfa297567c353446506ee4b4e39c84454b09"
      }
    }
  },
  "classifications":[
    {
      "name":"ClamAV",
      "scanned":"2016-08-30T06:31:48Z",
      "classification":"Win.Spyware.SpyEyes-94"
    }
```

```
  ]
}
```

2.3. Malware & Infrastructure Relationships

The STIX Malware and Infrastructure SDOs are interrelated and should be developed in tandem - see diagram below.



2.4. Malware (merged)

Type Name: `malware`

This SDO represents a merged version of the Malware Instance and Malware (family) objects defined below in sections 2.6 and 2.7, respectively.

Open Questions:

1. What are the use cases that we want to solve?
    a. Is there a priority? Should we do all for 2.1 or just some?
    b. Notional use cases:
        i. Use in indicators for cyber defense/response
            1. Indicator -> Malware Family -> COA
        ii. Use in CTI to help analysts
            1. Use in building finished intel (e.g. to determine relevance, attribution)
            2. Tracking outbreaks and activity

        a. Indicator -> Malware / Family
        b. Sighting -> Indicator (or Malware potentially)
        c. Event/Incident -> Indicator/Malware
    iii. Representing static/dynamic analysis results
        1. Run X sandbox, dump output to malware object
        2. Indicator extraction (IOCs)
        3. Behavioral characterization
        4. Shared analysis
  c. Discuss at May 16 working call, continue at F2F. Objectives:
    i. Identify whether there are obvious use cases that we missed.
    ii. Assess whether there are use cases that are mischaracterized or redundant.
    iii. Assess whether there's a priority that would let us produce an "MVP" in 2.1 and then expand in future releases.
2. Is there one Malware object or a Malware Instance object and a Malware (family) object?
  a. Proceed to develop with unified object, when we have a final proposal we can weigh in if there's a compelling reason to change.
3. How much capability do you put in the "strings" field?
  a. Is regex and Yara only for families?
  b. If we choose to capture this, how do we do it?
  c. Discuss at F2F
4. How do you treat targeting?
  a. Technical targeting by OS, software, architecture
    i. Vulnerabilities
  b. Information gathering targeting by data type (what types of information does it look for on a system).
    i. .docx, .pptx
    ii. PII, PHI, etc.
  c. Victim targeting by industry, location, etc.
  d. Discuss at F2F
5. How much detail do we want to capture in action?
  a. Full action model (discussion TBD)
  b. Some subset of analysis results via specific properties:
    i. dropped_files
    ii. domain_lookups
    iii. Connected_ip_addresses
  c. Discuss at F2F
6. Do you include the actual sample as an artifact? A URL to a sample?
  a. What is the use case?
  b. How do you allow for .zip and password protection so that it can make its way through firewalls and other detections?
  c. Discuss at F2F
7. What do we want to do for the labels vocabulary?
  a. Do we greatly expand it? Do we leave it fairly limited and make minor modifications?
  b. Discuss at F2F
8. How do we avoid people getting confused as to whether they should use an Indicator, a Malware, or both (linked by an indicates relationship.)
  a. We will add text to the description for the SDO that tries to clarify this. We'll need to evaluate after that's done to make sure that works.
  b. We may also need best-practice guidance.

2.4.1. Properties

Common Properties
type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_markings_refs, granular_markings
Malware Specific Properties
is_family, name, description, kill_chain_phases, first_seen, last_seen, targeted_operating_systems, certificates, strings, actions, sample_metadata, sample, extra_analysis_data, classifications

Property Name	Type	Description
type (required)	`string`	The value of this field **MUST** be `malware`
labels (required)	`list` of type `open-vocab`	The type of malware being described. This is an open vocabulary and values **SHOULD** come from the `malware-labels-ov` vocabulary.
external_references (optional)	`list` of type `external-reference`	A list of external references which refer to non-STIX information. This field **MAY** be used to capture names for this malware instance across vendors and organizations. When doing so, the **source** property **SHOULD** be used to capture the vendor, organization, or tool name and the **external_id** property **SHOULD** be used to capture the exact name it's known by. For example, to capture that ACME Inc. calls this malware instance "foobar", an external reference could be added with a **source** of `acme-inc` and an **external_id** of `foobar`. This field **SHOULD NOT** be used to capture AV classifications.
is_family (required)	`boolean`	Specifies whether the object represents a Malware Family (if

		`true`) or a Malware Instance (if `false`).
`name` (required)	`string`	A name used to identify the Malware Instance or Family, as specified by the producer of the SDO. If a name for a Malware Instance is not available, the SHA-256 hash value or binary filename **SHOULD** be used instead.
`description` (optional)	`string`	A description that provides more details and context about the Malware Instance or Family, potentially including its purpose and its key characteristics.
`kill_chain_phases` (optional)	`list` of type `kill-chain-phase`	The list of Kill Chain Phases for which this Malware Instance or Family can be used.
`first_seen` (optional)	`timestamp`	The time that the Malware Family or Malware Instance was first seen.
`last_seen` (optional)	`timestamp`	The time that the Malware Family or Malware Instance was last seen.
`targeted_operating_systems` (optional)	`?`	Specifies the operating systems targeted by the Malware Family (i.e., the set of operating systems known to be targeted by the family) or Malware Instance.
`targeted_architecture` (optional)	`open-vocab`	Specifies the processor architecture(s) (e.g., x86, ARM, etc.) targeted by the Malware Family or Malware Instance.
`certificates` (optional)	`observable-objects`	Specifies either: ● For a Malware Family (**`is_family`** = `true`), the certificates that are common to the family ● For a Malware Instance (**`is_family`** = `false`), the certificates used to sign the binary associated with the instance or to encrypt its network traffic One and only one Cyber Observable Object in this container **MUST** be of type `x509-certificate`. Any

		additional Cyber Observable Objects in this container **MUST** be referenced by this object.
strings (optional)	`list` of type `string`	Specifies either: • For a Malware Family (**is_family** = `true`), the strings that are common to the family • For a Malware Instance (**is_family** = `false`), the strings extracted from the binary associated with the instance
code_snippets (optional)	`observable-objects`	Specifies either: • For a Malware Family (**is_family** = `true`), one or more code snippets that are common to the family • For a Malware Instance (**is_family** = `false`), one or more code snippets extracted from the binary associated with the instance Each Cyber Observable Object in this container **MUST** be of type `artifact`.
network_traffic (optional)	`observable-objects`	Specifies either: • For a Malware Family (**is_family** = `true`), the network traffic that is common to the family • For a Malware Instance (**is_family** = `false`), the network traffic that was observed during the instrumented execution of the instance The root Cyber Observable Object in the `observable-objects` container **MUST** be of type `network-traffic`.
actions (optional)	`list` of type `observables/action`	Specifies either: • For a Malware Family (**is_family** = `true`), the Actions that are common to the family • For a Malware Instance (**is_family** = `false`), the

		actions observed during the instrumented execution of the instance
sample_metadata (optional)	observable-objects	Specifies any metadata extracted from the binary associated with the Malware Instance, such as file headers. You must not use any reference fields on the object <TODO: add additional text on which properties are invalid> This property **MUST** be included if **is_family** is set to false and **MUST NOT** be used if **is_family** is set to true. The root Cyber Observable Object in this container **MUST** be of type file.
sample (optional)	observable-objects	Specifies the actual binary of the Malware Instance as a base64-encoded payload. <TODO: add additional text on which properties are invalid> This property **MUST NOT** be used if **is_family** is set to true. The root Cyber Observable Object in this container **MUST** be of type artifact.
extra_analysis_data (optional)	dictionary	Specifies any additional analysis data observed for the Malware Instance, as a set of key/value pairs. This property **MUST NOT** be used if **is_family** is set to true.
classifications (optional)	list of type classifications-type	Specifies any scan data or classifications captured for the Malware Instance. This property **MUST NOT** be used if **is_family** is set to true.

2.4.2. Classifications Type

Type Name: classifications-type

The Classifications Type captures classification data as reported by antivirus (AV) and similar types of tools.

Summary		
product, engine_version, definition, scanned, classification, details		
Property Name	**Type**	**Description**
product (required)	open-vocab	This property captures the name of the AV engine or product that was used. Product names **SHOULD** be all lower-case with words separated by a dash "-". See av-product-ov
engine_version (optional)	string	This property captures the version of the AV engine used by the AV scanner tool. Example: "1.3.2.4r1234"
definition (optional)	string	This property captures the version of the AV definitions used by the AV scanner tool. Example: "1.3.2.4r1234"
submitted (optional)	timestamp	This property specifies the date and time that the malware was first submitted for scanning. This value will stay constant while the scanned date can change.
scanned (optional)	timestamp	This property specifies the date and time of the scan. This field can be used to capture how a scan changes over time.
classification (optional)	string	This property captures the classification or name assigned to the malware instance by the AV scanner tool. If the classification property is omitted, the tool did not classify this Malware Instance as malicious. Example: Win.Spyware.SpyEyes-94
details (optional)	string	This property can capture any extra notes or details from the scan.

2.4.3. Vocabularies

2.4.3.1. Malware Labels Vocabulary

Vocabulary Name: malware-labels-ov

An open vocabulary of malware labels.

Value	Description
`adware`	Specifies any software that is funded by advertising. Some adware may install itself in such a manner as to become difficult to remove, hiding components and disabling removal techniques. Adware may also gather sensitive user information from a system.
`appender`	Specifies a file-infecting virus that places its code at the end of the files it infects, adjusting the file's entry point to cause its code to be executed before that of the original file.
`backdoor`	Specifies a piece of software which, once running on a system, opens a communication vector to the outside so that the computer can be accessed remotely by an attacker.
`boot-sector-virus`	Specifies a virus that infects the master boot record of a storage device.
`bot`	Specifies a program which resides on an infected system, communicating with and forming part of a botnet. The bot may be implanted by a worm or trojan, which opens a backdoor. The bot then monitors the backdoor for further instructions.
`clicker`	Specifies a trojan that makes a system visit a specific web page, often very frequently and usually with the aim of increasing the traffic recorded by the site and thus increasing revenue from advertising. Clickers may also be used to carry out DDoS attacks.
`companion-virus`	Specifies a virus that takes the place of a particular file on a system instead of injecting code into it.
`cavity-filler`	Specifies a type of file-infecting virus which seeks out unused space within the files it infects, inserting its code into these gaps to avoid changing the size of the file and thus not alerting integrity-checking software to its presence.
`data-diddler`	Specifies a type of malware that makes small, random changes to data, such as data in a spreadsheet, to render the data contained in a document inaccurate and in some cases worthless.
`downloader`	Specifies a small trojan file programmed to download and execute other files, usually more complex malware.
`dropper-file`	Specifies a type of Trojan that deposits an enclosed payload onto a destination host computer by loading itself into memory, extracting the malicious payload, and then writing it to the file system..
`file-infector-virus`	Specifies a virus that infects a system by inserting itself somewhere in existing files; this is the "classic" form of virus.
`fork-bomb`	Specifies a very simple form of malware, a type of rabbit which simply launches more copies of itself. Once a fork bomb is executed, it will attempt to run several

	identical processes, which will do the same, the number growing exponentially until the system resources are overwhelmed by the number of identical processes running, which may in some cases bring the system down and cause a denial of service.
`greyware`	Specifies software that, while not definitely malicious, has a suspicious or potentially unwanted aspect.
`implant`	Specifies code inserted into an existing program using a code patcher or other tool.
`infector`	Specifies a function of malware that alters target files for the purpose of persisting and hiding the injected malware.
`keylogger`	Specifies a type of program implanted on a system to monitor the keys pressed and thus record any sensitive data, such as passwords, entered by the user.
`kleptographic-worm`	Specifies a worm that encrypts information assets on compromised systems so they can only be decrypted by the worm's author, also known as information-stealing worm.
`macro-virus`	Specifies a virus that uses a macro language, for example in Microsoft Office documents.
`malcode`	Short for malicious code, also known as malware.
`mass-mailer`	Specifies a worm that uses email to propagate across the internet.
`metamorphic_virus`	Specifies a virus that changes its own code with each infection.
`mid-infector`	Specifies a type of file-infecting virus which places its code in the middle of files it infects. It may move a section of the original code to the end of the file, or simply push the code aside to make space for its own code.
`mobile-code`	Specifies 1. Code received from remote, possibly untrusted systems, but executed on a local system. 2. Software transferred between systems (e.g across a network) and executed on a local system without explicit installation or execution by the recipient.
`multipartite-virus`	Specifies malware that infects boot records, boot sectors, and files.
`password-stealer`	Specifies a type of trojan designed to steal passwords, personal data and details, or other sensitive information from the infected system.
`polymorphic-virus`	Specifies a type of virus that encrypts its code differently with each infection, or generation of infections.
`premium-dialer-smser`	Specifies a piece of malware whose primary aim is to dial or send SMS messages to premium rate numbers.
`prepender`	Specifies a file-infecting virus which inserts code at the beginning of the files it infects.

`ransomware`	Specifies a type of malware that encrypts files on a victim's system, demanding payment of ransom in return for the access codes required to unlock files.
`rat`	Specifies a remote access trojan or RAT, which is a trojan horse capable of controlling a machine through commands issue by a remote attacker.
`rogue-anti-malware`	Specifies a fake security product that demands money to clean phony infections.
`rootkit`	Specifies that the malware hides files or processes from normal methods of monitoring to conceal its presence and activities. Originally, the term applied to UNIX-based operating systems - a root kit was a collection of tools to enable a user to obtain root (administrator-level) access to a system and conceal any changes they might make. Such tools often included trojanized versions of standard monitoring software which would hide the root kit operators' activities. More recently the term has generally been applied to malware using stealth techniques. Rootkits can operate at a number of levels, from the application level - simply replacing or adjusting the settings of system software to prevent the display of certain information - through hooking certain functions or inserting modules or drivers into the operating system kernel, to the deeper level of firmware or virtualization rook kits, which are activated before the operating system and thus even harder to detect while the system is running.
`shellcode`	Specifies 1. A small piece of code that activates a command-line interface to a system that can be used to disable security measures, open a backdoor, or download further malicious code. 2. A small piece of code that opens a system up for exploitation, sometimes by not necessarily involving a command-line shell.
`spaghetti-packer`	Specifies that the malware uses a packer that obfuscates programs by emitting "spaghetti" code with a complex and tangled control structure.
`spyware`	Specifies software that gathers information and passes it to a third-party without adequate permission from the owner of the data. It may also be used in a wider sense, to include software that makes changes to a system or any of its component software, or which makes use of system resources without the full understanding and consent of the system owner.
`trojan-horse`	Specifies a piece of malicious code disguised as something inert or benign.
`variant`	The 'variant' value refers to the fact that types of malware can be subdivided into a number of families, or groups sharing many similarities, generally based on the same blocks of code and sharing similar behaviours. Within a family, a variant signifies a single individual item that is uniquely different from other members of the same family.
`virus`	Specifies 1. A self-replicating malicious program that requires human interaction to replicate. 2. A self-replicating program that runs and spreads by modifying other programs or files.
`wabbit`	Specifies a form of self-replicating malware that makes copies of itself on the local system. Unlike worms, rabbits do not attempt to spread across networks.
`web-bug`	Specifies a piece of code, generally a small file such as a tiny, transparent GIF

	image, which is used to track data on those viewing the page or mail in which it is hidden.
`wiper`	Specifies a piece of malware whose primary aim is to delete files or entire disks on a machine.
`worm`	Specifies 1. A self-replicating malicious program that replicates using a network and does not require human interaction. 2. A self-replicating, self-propagating, self-contained program that uses networking mechanisms to spread itself.
`zip-bomb`	Specifies a file compressed into some archive format and that expands to an enormous size when uncompressed, often by looping over the extraction code until the system's resources are exhausted.
`unknown`	Specifies that while the thing being characterized is probably malware, there is insufficient data as yet to justify applying any other labels from this vocabulary.

2.4.4. Relationships

These are the relationships explicitly defined between the Malware object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from this object by way of the Relationship Object. The reverse relationships (relationships "to" this object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship name or, as with open vocabularies, user-defined names.

Embedded Relationships	
`created_by_ref`	`source`
`object_markings_refs`	`marking-definition`

Common Relationships			
`duplicate-of`, `derived-from`, `related-to`			
Source	**Name**	**Target**	**Description**
`malware`	`authored-by`	`threat-actor`, `intrusion-set`	The Malware Instance or Malware Family was developed by the related threat actor or intrusion set.
`malware`	`delivered-by`	`attack-pattern`, `infrastructure`, `tool`	This covers the Malware Instance or Malware Family being delivered by malicious infrastructure such as exploit

			kits or by attack patterns such spear phishing.
`malware`	`targets`	`identity`, `vulnerability`	This Relationship documents that this Malware is being used to target this Victim Target or exploit the Vulnerability. For example, a `targets` Relationship linking a Malware representing a downloader to a Vulnerability for CVE-2016-0001 means that the malware exploits that vulnerability. Similarly, a `targets` Relationship linking a Malware representing a downloader to an Identity representing the energy sector means that downloader is typically used against targets in the energy sector.
`malware`	`uses`	`infrastructure`, `tool`	This Relationship documents that this Malware uses the related infrastructure or tool to perform its functions. For example, a `uses` Relationship linking a Malware representing a trojan to an Infrastructure representing a command and control botnet means that the trojan uses the botnet for C2.
`malware`	`variant-of`	`malware`	This Relationship is used to document that one Malware Instance or Family is a variant of another Malware Instance or Family. Only the following uses of this relationship are valid: Malware (**`is_family`** = `false`) → Malware (**`is_family`** = `true`): a Malware Instance is a variant

			of a Malware Family. For example, a particular Zeus version 2 sample is a variant of the broader Zeus family. Malware (**is_family** = `true`) → Malware (**is_family** = `true`): a Malware Family is a variant of another Malware Family. For example, the Gameover Zeus family is a variant of the broader Zeus family. Malware (**is_family** = `false`) → Malware (**is_family** = `false`): a Malware Instance is a variant of another Malware Instance. For example, a Malware Instance is a packed variant of another Malware Instance.
`malware`	`dropped-by`	`malware`	This Relationship covers the case where a Malware Instance drops another Malware Instance or a tool. This is especially common with “first-stage” Malware Instances such as downloaders.
Reverse Relationships			
`indicator`	`indicates`	`malware`	See forward relationship for definition.
`course-of-action`	`mitigates`	`malware`	See forward relationship for definition.
`attack-pattern`, `campaign`, `intrusion-set`, `threat-actor`	`uses`	`malware`	See forward relationship for definition.

2.4.5. Examples

Basic Malware Family

```
{
  "type": "malware",
  "id": "malware--16f4f3f9-1b68-4abb-bb66-7639d49f1e30",
  "created": "2016-05-21T08:17:27.000000Z",
```

```
  "modified": "2016-05-21T08:17:27.000000Z",
  "is_family": true,
  "labels":[
    "trojan"
  ],
  "name": "Gameover Zeus",
  "strings": [
      "tellerplus",
      "silverlake",
      "fdmaster.exe"
  ]
}
```

Related (dropped by) Malware Families

```
{
  "type": "malware",
  "id": "malware--16f4f3f9-1b68-4abb-bb66-7639d49f1e30",
  "created": "2016-05-21T08:17:27.000000Z",
  "modified": "2016-05-21T08:17:27.000000Z",
  "is_family": true,
  "labels":[
    "trojan"
  ],
  "name": "Gameover Zeus",
  "strings": [
      "tellerplus",
      "silverlake",
      "fdmaster.exe"
  ]
}

{
  "type": "relationship",
  "id": "relationship--0d574df9-a605-4d3c-9459-f736779dd040",
  "created": "2016-05-23T08:17:27.000000Z",
  "modified": "2016-05-23T08:17:27.000000Z",
  "relationship_type": "dropped-by",
  "source_ref": "malware--0d574df9-a605-4d3c-9459-f736779dd040"
  "target_ref": "malware--16f4f3f9-1b68-4abb-bb66-7639d49f1e30"
}

{
  "type": "malware",
  "id": "malware--0d574df9-a605-4d3c-9459-f736779dd040",
  "created": "2016-05-22T08:17:27.000000Z",
  "modified": "2016-05-22T08:17:27.000000Z",
  "is_family": true,
  "labels":[
    "ransomware"
  ],
  "name": "Cryptolocker",
  "external_references": [
        "Ransom.Cryptolocker",
        "Trojan.Gpcoder"
   ]
}
```

Basic Malware Instance (embedded Cyber Observables approach)

```
{
  "type":"malware",
  "id":"malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
```

```
  "created":"2016-05-12T08:17:27.000000Z",
  "modified":"2016-05-12T08:17:27.000000Z",
  "name":"SpyEye",
  "is_family":false,
  "labels":[
    "trojan"
  ],
  "sample_metadata":{
    "0":{
      "type":"file",
      "name":"cleansweep.exe",
      "size":126464,
      "hashes":{
        "MD5":"84714c100d2dfc88629531f6456b8276",
        "SHA-256":"861aa9c5ddcb5284e1ba4e5d7ebacfa297567c353446506ee4b4e39c84454b09"
      }
    }
  },
  "classifications":[
    {
      "name":"ClamAV",
      "scanned":"2016-08-30T06:31:48Z",
      "classification":"Win.Spyware.SpyEyes-94"
    }
  ]
}
```


Malware Instance w/ General Analysis Data (embedded Cyber Observables approach)


```
{
  "type":"malware",
  "id":"malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created":"2016-05-12T08:17:27.000000Z",
  "modified":"2016-05-12T08:17:27.000000Z",
  "is_family":false,
  "name":"SpyEye",
  "labels":[
    "trojan"
  ],
  "sample_metadata":{
    "0":{
      "type":"file",
      "name":"cleansweep.exe",
      "hashes":{
        "MD5":"84714c100d2dfc88629531f6456b8276"
      },
      "size":126464,
      "extensions":{
        "windows-pebinary-ext":{
          "pe_type":"exe",
          "imphash":"c0249a6a0570c835b3a4e210b910a600",
          "time_date_stamp":"2010-02-08T15:30:55Z",
          "sections":[
            {
              "name":".text",
              "entropy":6.25
            },
            {
              "name":".rdata",
              "entropy":2.58
```

```
            },
            {
              "name":".data",
              "entropy":6.14
            },
            {
              "name":".rsrc",
              "entropy":3.82
            },
            {
              "name":".reloc",
              "entropy":1.72
            }
          ]
        }
      }
    }
  },
  "strings":[
    "cleansweep",
    "strlen"
  ],
  "classifications":[
    {
      "name":"ClamAV",
      "scanned":"2016-08-30T06:31:48Z",
      "classification":"Win.Spyware.SpyEyes-94"
    }
  ]
}
```

Malware Instance w/ Dynamic Analysis Data (property-based approach)

```
{
  "type":"malware",
  "id":"malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created":"2016-05-12T08:17:27.000000Z",
  "modified":"2016-05-12T08:17:27.000000Z",
  "is_family":false,
  "name":"SpyEye",
  "labels":[
    "trojan"
  ],
  "sample_metadata":{
    "0":{
      "type":"file",
      "name":"cleansweep.exe",
      "hashes":{
        "MD5":"84714c100d2dfc88629531f6456b8276"
      },
      "size":126464,
      "extensions":{
        "windows-pebinary-ext":{
          "pe_type":"exe",
          "imphash":"c0249a6a0570c835b3a4e210b910a600",
          "time_date_stamp":"2010-02-08T15:30:55Z",
          "sections":[
            {
              "name":".text",
              "entropy":6.25
```

```
            },
            {
              "name":".rdata",
              "entropy":2.58
            },
            {
              "name":".data",
              "entropy":6.14
            },
            {
              "name":".rsrc",
              "entropy":3.82
            },
            {
              "name":".reloc",
              "entropy":1.72
            }
          ]
        }
      }
    }
  },
  "network_traffic":[
    {
      "0":{
        "type":"network-traffic",
        "dst_ref":"1",
        "protocols":[
          "tcp"
        ]
      },
      "1":{
        "type":"ipv4-addr",
        "value":"198.51.100.3"
      }
    },
    {
      "0":{
        "type":"network-traffic",
        "dst_ref":"1",
        "protocols":[
          "tcp"
        ]
      },
      "1":{
        "type":"ipv4-addr",
        "value":"198.51.100.27"
      }
    }
  ]
}
```

Malware Instance w/ Dynamic Analysis Data (Actions approach)

```
{
  "type":"malware",
  "id":"malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created":"2016-05-12T08:17:27.000000Z",
  "modified":"2016-05-12T08:17:27.000000Z",
  "is_family":false,
  "name":"SpyEye",
```

```
  "labels":[
    "trojan"
  ],
  "sample_metadata":{
    "0":{
      "type":"file",
      "name":"cleansweep.exe",
      "hashes":{
        "MD5":"84714c100d2dfc88629531f6456b8276"
      },
      "size":126464,
      "extensions":{
        "windows-pebinary-ext":{
          "pe_type":"exe",
          "imphash":"c0249a6a0570c835b3a4e210b910a600",
          "time_date_stamp":"2010-02-08T15:30:55Z",
          "sections":[
            {
              "name":".text",
              "entropy":6.25
            },
            {
              "name":".rdata",
              "entropy":2.58
            },
            {
              "name":".data",
              "entropy":6.14
            },
            {
              "name":".rsrc",
              "entropy":3.82
            },
            {
              "name":".reloc",
              "entropy":1.72
            }
          ]
        }
      }
    }
  },
  "actions":[
    {
      "0":{
        "type":"action",
        "name":"create file",
        "output_object_refs":[
          "2"
        ],
        "timestamp":"2016-01-20T12:31:12.12345Z"
      },
      "1":{
        "type":"directory",
        "path":"C:\\Windows\\System32"
      },
      "2":{
        "type":"file",
        "hashes":{
          "SHA-256":"ceafbfd424be2ca4a5f0402cae090dda2fb0526cf521b60b60077c0f622b285a"
        },
```

```
        "parent_directory_ref":"1",
        "name":"qwerty.dll"
      }
    },
    {
      "0":{
        "type":"action",
        "name":"establish tcp connection",
        "output_object_refs":[
          "1"
        ],
        "timestamp":"2016-01-20T12:33:04.12345Z"
      },
      "1":{
        "type":"network-traffic",
        "dst_ref":"2",
        "protocols":[
          "tcp"
        ]
      },
      "2":{
        "type":"ipv4-addr",
        "value":"198.51.100.3"
      }
    },
    {
      "0":{
        "type":"action",
        "name":"establish tcp connection",
        "output_object_refs":[
          "1"
        ],
        "timestamp":"2016-01-20T12:33:53.12345Z"
      },
      "1":{
        "type":"network-traffic",
        "dst_ref":"2",
        "protocols":[
          "tcp"
        ]
      },
      "2":{
        "type":"ipv4-addr",
        "value":"198.51.100.27"
      }
    }
  ]
}
```

Multiple Malware Instances Belonging to the Same Family

```
{
  "type":"malware",
  "id":"malware--109fc567-307f-4efa-85f5-2398183f09c3",
  "created":"2016-05-22T08:17:27.000000Z",
  "modified":"2016-05-22T08:17:27.000000Z",
  "is_family":false,
  "sample_metadata":{
    "0":{
      "hashes":{
        "MD5":"bc11c93f1b6dc74bf4804a35b34d9267",
```

```
        "SHA-256":"a2bc3059283d7cc7bc574ce32cb6b8bfd27e02ac3810a21bd3a9b84c17f18a72"
      }
    }
  }
}

{
  "type":"malware",
  "id":"malware--1d43ff49-daca-454a-9025-562ac4ad9321",
  "created":"2016-05-22T08:17:27.000000Z",
  "modified":"2016-05-22T08:17:27.000000Z",
  "is_family":false,
  "sample_metadata":{
    "0":{
      "hashes":{
        "MD5":"b17603f401719f1d99ad6472f8d6682a",
        "SHA-256":"0be1f445537f124b5175e1f2d1da87e2e57aa4ba09ea5fe72b7bafaf0b8f9ad2"
      }
    }
  }
}

{
  "type":"malware",
  "id":"malware--f8cc9d93-5455-4ac7-9f1f-0abff9a65f2e",
  "created":"2016-05-22T08:17:27.000000Z",
  "modified":"2016-05-22T08:17:27.000000Z",
  "is_family":false,
  "sample_metadata":{
    "0":{
      "hashes":{
        "MD5":"f1e2de2a9135138ef5b15093612dd813",
        "SHA-256":"136e8991816b958bb76aaf22fefd18194cf78a80e95d572754f95e1f86149a65"
      }
    }
  }
}

{
  "type": "malware",
  "id": "malware--0d574df9-a605-4d3c-9459-f736779dd040",
  "created": "2016-05-22T08:17:27.000000Z",
  "modified": "2016-05-22T08:17:27.000000Z",
  "is_family": true,
  "name": "Cryptolocker",
  "external_references": [
        "Ransom.Cryptolocker",
        "Trojan.Gpcoder"
   ]
}

{
  "type": "relationship",
  "id": "relationship--f9618034-8c76-4406-baa5-699d319f24d0",
  "created": "2016-05-23T08:17:27.000000Z",
  "modified": "2016-05-23T08:17:27.000000Z",
  "relationship_type": "variant-of",
  "source_ref": "malware--109fc567-307f-4efa-85f5-2398183f09c3"
```

```
  "target_ref": "malware--0d574df9-a605-4d3c-9459-f736779dd040"
}

{
  "type": "relationship",
  "id": "relationship--43da1a19-8c5f-47f2-a09b-c9dd1ab58dd8",
  "created": "2016-05-23T08:17:27.000000Z",
  "modified": "2016-05-23T08:17:27.000000Z",
  "relationship_type": "variant-of",
  "source_ref": "malware--1d43ff49-daca-454a-9025-562ac4ad9321"
  "target_ref": "malware--0d574df9-a605-4d3c-9459-f736779dd040"
}

{
  "type": "relationship",
  "id": "relationship--1a965c7f-166e-432b-b426-ced1ecf4d7de",
  "created": "2016-05-23T08:17:27.000000Z",
  "modified": "2016-05-23T08:17:27.000000Z",
  "relationship_type": "variant-of",
  "source_ref": "malware--f8cc9d93-5455-4ac7-9f1f-0abff9a65f2e"
  "target_ref": "malware--0d574df9-a605-4d3c-9459-f736779dd040"
}
```


Related Malware Instances (dropped)


```
{
  "type":"malware",
  "id":"malware--abffe0ac-5a3b-4757-89e5-9756885d7601",
  "created":"2016-05-21T08:17:27.000000Z",
  "modified":"2016-05-21T08:17:27.000000Z",
  "is_family":false,
  "name":"Gameover Zeus v3",
  "sample_metadata":{
    "0":{
      "hashes":{
        "MD5":"5e5e46145409fb4a5c8a004217eef836",
        "SHA-256":"3ff49706e78067613aa1dcf0174968963b17f15e9a6bc54396a9f233d382d0e6"
      }
    }
  }
}


{
  "type": "relationship",
  "id": "relationship--81055dc2-e5a5-471a-ac16-52df1b714ff7",
  "created": "2016-05-23T08:17:27.000000Z",
  "modified": "2016-05-23T08:17:27.000000Z",
  "relationship_type": "dropped-by",
  "source_ref": "malware--109fc567-307f-4efa-85f5-2398183f09c3"
  "target_ref": "malware--abffe0ac-5a3b-4757-89e5-9756885d7601"
}

{
  "type":"malware",
  "id":"malware--109fc567-307f-4efa-85f5-2398183f09c3",
  "created":"2016-05-22T08:17:27.000000Z",
  "modified":"2016-05-22T08:17:27.000000Z",
  "is_family":false,
  "name":"Cryptolocker",
```

```
  "sample_metadata":{
    "0":{
      "hashes":{
        "MD5":"bc11c93f1b6dc74bf4804a35b34d9267",
        "SHA-256":"a2bc3059283d7cc7bc574ce32cb6b8bfd27e02ac3810a21bd3a9b84c17f18a72"
      }
    }
  }
}
```

2.5. Intel Note

Type Name: `intel-note`	**Status: Status:** Development

An Intel Note is a comment or note containing informative text to help explain the context of one or more STIX Objects (SDOs or SROs) or to provide additional analysis that is not contained in the original object. Intel Notes can be created by either the original object creator of the objects it relates to or by others.

For example, an analyst may add a note to a Campaign object created by another organization indicating that they've seen posts related to that campaign on a hacker forum.

Because Intel Notes are typically (though not always) created by human analysts and are comprised of human-oriented text, they contain an additional property to capture the analyst that created the note. This is distinct from the **`created_by_ref`** property, which is meant to capture the organization that created the object.

Properties

Common Properties		
<TODO>		
Intel Note Specific Properties		
`name, description, author, object_refs`		
Property Name	**Type**	**Description**
`type` (required)	`string`	The value of this field **MUST** be `intel-note`
`name` (required)	`string`	A name used to identify the Intel Note as a summary of the note
`description` (required)	`string`	The content of the note.
`author` (optional)	`string`	The name of the author of this note (e.g., the analyst that created it).

object_refs (required)	list of type identifier	The STIX Objects (SDOs and SROs) that the note is being applied to.

Relationships

There are no relationships between the Intel Note object and other objects, other than the embedded relationships listed below by property name along with their corresponding target.

Embedded Relationships	
created_by_ref	identity
object_marking_refs	marking-definition
author_ref	identifier (of type Identity)
object_refs	list of type identifier (of type any STIX Object type)

Examples

A generic Intel Note defining additional context and shows an optional external reference to a ticketing system.

```
{
  "type": "intel-note",
  "id": "intel-note--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created": "2016-05-12T08:17:27.000Z",
  "modified": "2016-05-12T08:17:27.000Z",
  "external_references": [
    {
      "source_name": "job-tracker",
      "id": "job-id-1234"
    }
  ],
  "name": "Tracking Team Note#1",
  "description": "This note indicates the various steps taken by the threat analyst team to investigate this specific incident. Step 1) Do a scan 2) Review scanned results for identified hosts not known by external intel….etc",
  "author_ref": "identity--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
  "object_refs": ["incident--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f"]
}
```

2.6. Opinion

Type Name: opinion	**Status:** Proposal **MVP :** Yes

An Opinion is an assessment of the correctness of the information in another STIX Object. The primary field is the **opinion** field, which captures the level of agreement or disagreement using a fixed scale. That fixed scale also supports a numeric mapping to allow for consistent statistical operations across opinions.

For example, an analyst from a consuming organization might say that they "strongly disagree" with a Campaign object and provide an explanation about why. In a more automated workflow, a SOC operator might give an indicator one star (expressing "strongly disagree") because it is a false positive.

Properties

Common Properties		
<TODO>		
Opinion Specific Properties		
object_ref, opinion		
Property Name	**Type**	**Description**
type (required)	`string`	The value of this field **MUST** be `opinion`
object_refs (required)	`list` of type `identifier`	The STIX Objects (SDOs and SROs) that the opinion is being applied to.
opinion (required)	`agreement-enum`	The opinion that the producer has about the object listed in the object_ref field. This is represented as an ordered vocabulary.
description (optional)	`string`	An explanation of why the creator has this opinion. For example, if an opinion of strongly-disagree is given, this can contain an explanation of why the object creator disagrees and what evidence they have for their disagreement.

Relationships

The Opinion object is a STIX Domain Object but MUST NOT have any SRO-based relationships to it or from it. It MUST have a direct embedded relationship to one STIX Object.

Embedded Relationships	
created_by_ref	`identity`
object_marking_refs	`marking-definition`
object_ref	`identifier` (of type any STIX Object type)

Examples

```
[
  {
    "type": "opinion",
    "id": "opinion--b01efc25-77b4-4003-b18b-f6e24b5cd9f7",
    "created": "2016-05-12T08:17:27.000Z",
    "modified": "2016-05-12T08:17:27.000Z",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "object_ref": "relationship--16d2358f-3b0d-4c88-b047-0da2f7ed4471",
    "opinion": "strongly-disagree",
    "description": "This doesn't seem like it is feasible. We've seen how PandaCat has
attacked Spanish infrastructure over the last 3 years, so this change in targeting seems too
great to be viable. The methods used are more commonly associated with the FlameDragonCrew."
  }
]
```

2.7. Agreement Vocab

Vocabulary Name: `agreement-enum`

The agreement vocabulary is currently used in the following SDOs:

- Comment

This vocabulary captures a degree of agreement with the information in a STIX Object. It is an ordered vocabulary, with the earlier terms representing disagreement, the middle term neutral, and the later terms representing agreement.

Vocabulary Summary	
`strongly-disagree`, `disagree`, `neutral`, `agree`, `strongly-agree`	
Vocabulary Value	**Description**
`strongly-disagree`	The creator strongly disagrees with the information and believes it is inaccurate or incorrect. This **MAY** be considered equivalent to a 1 in a numeric scale.
`disagree`	The creator disagrees with the information and believes it is inaccurate or incorrect. This **MAY** be considered equivalent to a 2 in a numeric scale.
`neutral`	The creator is neutral about the accuracy or correctness of the information. This **MAY** be considered equivalent to a 3 in a numeric scale.
`agree`	The creator agrees with the information and believes that it is accurate and correct.

	This **MAY** be considered equivalent to a 4 in a numeric scale.
`strongly-agree`	The creator strongly agrees with the information and believes that it is accurate and correct. This **MAY** be considered equivalent to a 5 in a numeric scale.

3. Patterning Changes

We are missing the following features:

- Case insensitive matching. This is required for matching Windows and MacOSX filenames.
- Matching substrings. For example, finding a file name in a registry key. If you know ahead of time what the file name is, it can be done, but if you pull a name from another object due to variables/backrefs, it cannot be done.
- Any additional functions (like string concatenation and or escaping)

3.1. Matches

The proposal is to slightly change how regex matches are handled. The proposal is that if a pattern has a MATCHES in it, and it gets repeated w/ the REPEAT operator, that any and all groups in the regex must be equal for all the observable data that matches.

For example, say you have the pattern `([ artifact:payload_bin MATCHES 'error: PAM: authentication error for [a-z]+ from 127.0.0.1 via 127.0.0.1' ] REPEATS 2 TIMES) WITHIN 10 SECONDS`. This would match any and all users, even if they were different users that failed to login. Under the proposal, if the pattern was `([ artifact:payload_bin MATCHES 'error: PAM: authentication error for ([a-z]+) from 127.0.0.1 via 127.0.0.1' ] REPEATS 2 TIMES) WITHIN 10 SECONDS` it would only match **IF** the user name was the same for all 5 log entries. So, the first would match w/ the following artifact logs, but the second would not match:

- `error: PAM: authentication error for abc from 127.0.0.1 via 127.0.0.1`
- `error: PAM: authentication error for def from 127.0.0.1 via 127.0.0.1`

3.2. Abstract Object Definitions

3.3. Variables/Backreferences

3.3.1. Declaring Variables

`[OBSERVATION EXPRESSION] AS X`

Where:

- `[OBSERVATION EXPRESSION]` is a valid Observation Expression that contains 1 or more Comparison Expressions that express some set of properties for the Cyber Observable Object that will be defined as a variable.
- `AS` is a new operator that assigns an Observation Expression to a variable name.
- `X` is a string that specifies the name of the variable that will be referenced later on in the pattern. Whitespace **MUST NOT** be included in the variable name.

3.3.2. Using Variables

```
[object:property_name = $X:property_name]
```

Where:

- `object:property_name` is the Object Path of another Cyber Observable Object.
- `$X` is a reference to a previously declared (using the `AS` operator) variable, with `$X:property_name` thereby referencing the value of a particular property on the Cyber Observable Object that is assigned to the variable.

3.3.3. Examples

The example below declares a File Object-based Observation Expression as a variable and then, in a separate Observation Expression, tests the value of a URL Object against the file name of the File Object denoted by the variable.

```
[(file:name = 'pdf.exe' OR file:size = '371712') AND file:created = t'2014-01-13T07:03:17Z']
AS PDFFILE FOLLOWED BY [ url:value LIKE $PDFFILE:name ]
```

3.4. External Lists

Propose alteration of the IN comparison operator to allow pointing at an external artifact object that contains either a CSV list or a JSON document containing a list

a `IN` (*x*,*y*,...),
 	
a `IN` {
`"url":"http://1.2.3.4/my_list.csv",`
`"hashes":{"MD5":"ABCDEF..."}`
}
 	
a `IN` {
`"url":"http://1.2.3.4/my_list.json"`
}
	a **MUST** be an Object Path and **MUST** evaluate to one of the values enumerated in a set (transitive).
	
	b may be either a set of predefined values encapsulated inside brackets (*x*,*y*,...), or an instance of a JSON object that refers to an external list of values to be enumerated.
	
	When *b* is an instance of a JSON object, the object MUST contain a "url" property. The value of this property MUST conform to RFC 1738. When retrieved, the document that the "url" property resolves to MUST be of either MIME type "text/csv" or "application/json". If the document is of MIME type "text/csv",
		`process:name IN ('proccy', 'proximus', 'badproc')`

	then each entry in the CSV document will be treated as a value in the set to be tested against. If the document is of MIME type "application/json", the document MUST contain a single JSON array, which contains the values in the set to be tested against. The JSON object MAY contain an optional property called "hashes", whose value is of type `hashes-type`, which contains hashes that MUST be used to validate the document retrieved if present. The set values in *b* **MUST** be constants of homogenous data type and **MUST** be valid data types for the Object Property specified by ***a.*** The return value is true if *a* is equal to one of the values in the list. If *a* is not equal to any of the items in the list, then the Comparison Expression evaluates to false.	

3.5. Functions

4. Changes to Existing Objects

4.1. Report (per MISP)

Changes from 2.0:

1. Published date was made optional, so you can iterate on unpublished reports and represent working documents.
2. Added a start date to track when the activity captured in the report first started.

Open Questions:

1. The MISP guys wanted to add a boolean field to indicate whether a report represents an incident. Do we want to do this? Seems to overlap with Event. We could always track whether a report includes incidents via the labels field. You could also ascertain whether a report includes incidents based on whether it includes any Event objects which have the is_incident boolean set to true.

Use Cases:

1. The original Report use case, which is to provide a STIX Bundle-like capability, but different insofar as semantically Bundle implies no context whereas Report does.
2. The "APT666" Report, with accompanying RSA press release use case.
3. A "working draft report", or contextual grouping of content that evolves over time.

Reports are collections of threat intelligence focused on one or more topics, such as a description of a threat actor, malware, or attack technique, including context and related details. Reports may be used to group related threat intelligence together, allowing the producer to publish the report as a as a comprehensive cyber threat story. Reports can also be used to convey a set of STIX SDOs and SROs which are contextually-bound, or simply used as a working document. The semantics of a report are defined by the information embedded within the report.

The Report SDO contains a list of references to SDOs and SROs along with a textual description and the name of the report.

For example, a threat report produced by ACME Defense Corp. discussing the Glass Gazelle campaign should be represented using Report. The Report itself would contain the narrative of the report while the Campaign SDO and any related SDOs (e.g., Indicators for the Campaign, Malware it uses, and the associated Relationships) would be referenced in the report contents.

For example, a report produced by a CERT can describe attempted or successful security events, intelligence analyses, take-down notification events, ongoing software analyses (e.g. pre-analysis), ongoing software vulnerability assessments. The report itself contains SDOs and SROs which can be generated from the above mentioned cases along with descriptive content.

4.1.1. Properties

Common Properties		
`type`, `id`, `created_by_ref`, `created`, `modified`, `revoked`, `labels`, `confidence`, `external_references`, `lang`, `object_marking_refs`, `granular_markings`		
Report Specific Properties		
`name`, `description`, `published`, `object_refs`		
Property Name	**Type**	**Description**
`type` (required)	`string`	The value of this property **MUST** be `report`
`labels` (required)	`list` of type `open-vocab`	This property is an Open Vocabulary that specifies the primary subject of this report. This is an open vocabulary and values **SHOULD** come from the `report-label-ov` vocabulary.
`name` (required)	`string`	A name used to identify the Report.
`description` (optional)	`string`	A description that provides more details and context about the

		Report, potentially including its purpose and its key characteristics.
published (optional)	`timestamp`	The date that this Report object was officially published by the creator of this report. The publication date (public release, legal release, etc.) may be different than the date the report was created or shared internally (the date in the **created** property). If the **published** property is absent, the report is unpublished.
object_refs (required)	`list` of type `identifier`	Specifies the STIX Objects that are referred to by this Report.
start (optional)	`timestamp`	The date describes when the subject of the Report object was officially detected, analysed or handled. The creation date may be different than the date of the Report object. In such a case, the **created** property should be set to a different timestamp.

4.1.2. Relationships

There are no relationships explicitly defined between the Report object and other objects, other than those defined as common relationships. The first section lists the embedded relationships by property name along with their corresponding target.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship name or, as with open vocabularies, user-defined names.

Embedded Relationships	
created_by_ref	`identifier` (of type `identity`)
object_marking_refs	`identifier` (of type `marking-definition`)
object_refs	`list` of type `identifier` (of STIX Object or

	`marking-definition` type)
Common Relationships	
`duplicate-of`, `derived-from`, `related-to`	

Examples

A standalone Report; the consumer may or may not already have access to the referenced STIX Objects.

```
{
  "type": "report",
  "id": "report--84e4d88f-44ea-4bcd-bbf3-b2c1c320bcb3",
  "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
  "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",
  "published": "2016-01-201T17:00:00Z",
  "labels": ["campaign"],
  "object_refs": [
    "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
    "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
    "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a"
  ]
}
```

A Bundle with a Report and the STIX Objects that are referred to by the Report

```
{
  "type": "bundle",
  "id": "bundle--44af6c39-c09b-49c5-9de2-394224b04982",
  "objects": [
    {
      "type": "identity",
      "id": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      ...,
      "name": "Acme Cybersecurity Solutions"
    },
    {
      "type": "report",
      "id": "report--84e4d88f-44ea-4bcd-bbf3-b2c1c320bcbd",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      "created": "2015-12-21T19:59:11.000Z",
      "modified": "2016-05-21T19:59:11.000Z",
      "name": "The Black Vine Cyberespionage Group",
      "description": "A simple report with an indicator and campaign",
      "published": "2016-01-201T17:00:00Z",
      "labels": ["campaign"],
      "object_refs": [
        "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
        "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
```

```
        "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a"
      ]
    },
    {
      "type": "indicator",
      "id": "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
      "created": "2015-12-21T19:59:17.000Z",
      "modified": "2016-05-21T19:59:17.000Z",
      "name": "Some indicator",
      "labels": ["malicious-activity"],
      "pattern": "[ file.hashes.MD5 = '3773a88f65a5e780c8dff9cdc3a056f3' ]",
      "valid_from": "2015-12-21T19:59:17Z",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283"
    },
    {
      "type": "campaign",
      "id": "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      "created": "2015-12-21T19:59:17.000Z",
      "modified": "2016-05-21T19:59:17.000Z",
      "name": "Some Campaign"
    },
    {
      "id": "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a",
      "type": "relationship",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      "created": "2015-12-21T19:59:17.000Z",
      "modified": "2015-12-21T19:59:17.000Z",
      "source_ref": "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
      "target_ref": "campaign--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
      "name": "indicates"
    }
  ]
}
```

An export event from a TIP platform (like MISP):

```
{
  "type": "report",
  "id": "report--59120865-27e0-4e6d-9b74-4a9f950d210f",
  "created_by_ref": "identity--55f6ea5e-2c60-40e5-964f-47a8950d210f",
  "created": "2017-05-09T19:59:11.000Z",
  "modified": "2017-05-09T19:59:11.000Z",
  "description": "Additional Analysis of EPS Processing Zero-Days Exploited by Multiple Threat
Actors",
  "published": "2017-05-10T17:00:00.000Z",
  "labels": ["ms-caro-malware:malware-platform=\"Win64\"",
"adversary:infrastructure-type=\"exploit-distribution-point\"",
"estimative-language:likelihood-probability=\"very-likely\""
],
  "object_refs": [
    "indicator--59120872-11dc-4982-8a6c-4c95950d210f",
    "vulnerability--5912097c-fc58-4e8e-bb3d-41d3950d210f",
    "relationship--59147a22-3100-4779-9377-360395ca48b7"
```

```
  ]
}
```

5. Vocabularies

5.1. Malware Label

Vocabulary Name: `malware-label-ov`

The malware label vocabulary is currently used in the following SDO(s):

- Malware

Malware label is an open vocabulary that represents different types and functions of malware. Malware labels are not mutually exclusive; a malware family or instance can be both spyware and a screen capture tool.

Vocabulary Summary	
`adware`, `backdoor`, `bot`, `ddos`, `dropper`, `exploit-kit`, `keylogger`, `ransomware`, `remote-access-trojan`, `resource-exploitation`, `rogue-security-software`, `rootkit`, `screen-capture`, `spyware`, `trojan`, `virus`, `worm`	
Vocabulary Value	**Description**
`adware`	Any software that is funded by advertising. Adware may also gather sensitive user information from a system.
`backdoor`	A malicious program that allows an attacker to perform actions on a remote system, such as transferring files, acquiring passwords, or executing arbitrary commands [Mell2005].
`bot`	A program that resides on an infected system, communicating with and forming part of a botnet. The bot may be implanted by a worm or Trojan, which opens a backdoor. The bot then monitors the backdoor for further instructions.
`ddos`	A tool used to perform a distributed denial of service attack.
`dropper`	A type of trojan that deposits an enclosed payload (generally, other malware) onto the target computer.
`exploit-kit`	A software toolkit to target common vulnerabilities.
`keylogger`	A type of malware that surreptitiously monitors keystrokes and either records them for later retrieval or sends them back to a central

	collection point.
`ransomware`	A type of malware that encrypts files on a victim's system, demanding payment of ransom in return for the access codes required to unlock files.
`remote-access-trojan`	A remote access trojan program (or RAT), is a trojan horse capable of controlling a machine through commands issued by a remote attacker.
`resource-exploitation`	A type of malware that steals a system's resources (e.g., CPU cycles), such as a bitcoin miner.
`rogue-security-software`	A fake security product that demands money to clean phony infections.
`rootkit`	A type of malware that hides its files or processes from normal methods of monitoring in order to conceal its presence and activities. Rootkits can operate at a number of levels, from the application level — simply replacing or adjusting the settings of system software to prevent the display of certain information — through hooking certain functions or inserting modules or drivers into the operating system kernel, to the deeper level of firmware or virtualization rootkits, which are activated before the operating system and thus even harder to detect while the system is running.
`screen-capture`	A type of malware used to capture images from the target systems screen, used for exfiltration and command and control.
`spyware`	Software that gathers information on a user's system without their knowledge and sends it to another party. Spyware is generally used to track activities for the purpose of delivering advertising.
`trojan`	Any malicious computer program which is used to hack into a computer by misleading users of its true intent.
`virus`	A malicious computer program that replicates by reproducing itself or infecting other programs by modifying them.
`worm`	A self-replicating, self-contained program that usually executes itself without user intervention.
`unknown`	There is not enough information available to determine the type of malware.