STIX Difficulties

A collection of things we’ve found difficult to do in STIX

Contents

[1. Object ID’s are optional but should be required 3](#_Toc435595494)

[2. Generate Object ID from hashed Object contents 4](#_Toc435595495)

[3. Deduplication is difficult 4](#_Toc435595496)

[4. There are too many ways to update an Object (Versioning) 7](#_Toc435595497)

[5. Observable Patterns and Observable Instances differences aren’t easily discerned 8](#_Toc435595498)

[6. Difficult to ask a request and get a response to that question 9](#_Toc435595499)

[7. Sightings are difficult to send out independently 10](#_Toc435595500)

[8. Cannot generate a relationship between two third-party objects 11](#_Toc435595501)

[9. Too many ways to do things 12](#_Toc435595502)

[10. Indicators/Observable, Composition/Object hierarchy is complicated 15](#_Toc435595503)

[11. Embedded and Referencing Objects 16](#_Toc435595504)

[12. Difficult for third parties to show agreement or disagreement 17](#_Toc435595505)

[13. Obfuscation of Producer **Content** is Difficult 18](#_Toc435595506)

[14. Obfuscation of Producer **Identity** is Difficult 19](#_Toc435595507)

[15. Difficult to ask for a referred object 20](#_Toc435595508)

[16. Victim Targeting is embedded within a TTP 20](#_Toc435595509)

[17. No STIX-wide way to handle aliases 21](#_Toc435595510)

[18. Interoperability is difficult if custom vocabularies are allowed 22](#_Toc435595511)

[19. Difficult to group 'possibly' related things during an investigation 22](#_Toc435595512)

[20. Cannot suggest hypotheses to a community through STIX 23](#_Toc435595513)

[21. Relationships are constrained to limited Objects within STIX 24](#_Toc435595514)

[22. TTPs are **almost** mandatory 25](#_Toc435595515)

[23. Which to use? Indicator Composition, Observable Composition, or referenced Object? 25](#_Toc435595516)

[24. Are CybOX <Object> IDs used in STIX? 26](#_Toc435595517)

[25. Some Object names are confusing 27](#_Toc435595518)

# Object ID’s are optional but should be required

## PROBLEM

Object IDs are optional at the moment. This restriction is fine if the object is always embedded within another Object, but this becomes problematic if the Object needs to be referenced at a later date. This could be due to a mistake being made and the wish to revoke something published, sudden awareness that a small outbreak is bigger than first thought (so need to reference Object originally embedded) or similar.

With the present optionality of Object ID it is difficult to always be sure that objects can be referenced in the future, which impacts their reusability.

## POTENTIAL ANSWER

A large number of the solutions proposed in this document rely on an Object having a referenceable the Object ID, that Object ID staying the same through the lifecycle of the Object, and the Object ID containing a namespace that resolves to a domain name in some way so that consumers can ‘lookup’ objects based on their IDs (if permitted).

Object IDs are essential to enable the piecing together of the sequence of events, or to do analysis focussing on the way that attackers operate. This is key to understanding the Tactics, Techniques and Procedures that attacker’s use, which in turn is highly useful for linking malware with the Threat Actors who create and use them. If a consumer does not require an Object ID they can simply ignore it. The impact of a consumer to not ingest the Object ID is far less than the impact of not having the Object ID in the first place.

There is one place that I’ve heard ID shouldn’t be required – and that is the STIX\_Package itself. Since STIX v1.2 the STIX\_Package is effectively a container that all the other STIX objects are contained within, then it is effectively a ‘throwaway’ object, meaning that we should require an ID if we are just throwing the object away. That said – if we do pursue the idea of the STIX Request/Response object as described in section 6 then we will need to identify the STIX Request so that the Responses can be matched to it – so maybe keeping STIX\_Packages with an ID is acceptable if we are allowing STIX Request/Responses.

# Generate Object ID from hashed Object contents

## PROBLEM

There has been a few people who have mentioned that they would like to create Object IDs from a hash of the Object contents. The argument is that this would help during deduplication of CybOX Objects, as the content would be the same if multiple Objects were detected, and duplication would be easy to detect.

## POTENTIAL ANSWER

Current list consensus seems to be that this should be permitted as a way of generating the ID, but that this shouldn’t be mandated as the only way that it is generated. It was posited that some lower powered devices may not have enough processing power to be able to generate a hash, and therefore mandating hash generation of the data would exclude them.

I believe that this should be mandated, as it provides a quick way of determining if the content was inadvertently modified during transit. As the hash is not a HMAC it does not provide malicious tampering detection (although this change would allow it to be supported in the future).

# Deduplication is difficult

## PROBLEM

Recognizing that two different Objects (with different Object IDs but both containing the same internal data) are related is currently outside the scope of STIX. This lack of guidance on how to detect duplication makes it difficult for implementers to actually build systems.

In most of the current Relationship fields (e.g. RelatedIncidentType, RelatedTTPType – all except RelatedObjectType) there isn’t a defined relationship vocabulary, meaning that there is no standard way for a producer to explicitly identify that another object is a duplicate of an Object. Even the RelatedObjectType which does have a default vocab doesn’t include the ability to describe duplicate objects within the options.

Using the following two objects as an example:

<cybox:Object id="myorg:Object-15be6630-c2df-4bf9-8750-3f45ca9e19cf">

<cybox:Properties xsi:type="AddressObj:AddressObjectType" category="ipv4-addr">

<AddressObj:Address\_Value>192.168.0.5</AddressObj:Address\_Value>

</cybox:Properties>

</cybox:Object>

<cybox:Object id="yourorg:Object-e2e89241-d858-4a29-b1ec-8155c3cd3278">

<cybox:Properties xsi:type="AddressObj:AddressObjectType" category="ipv4-addr">

<AddressObj:Address\_Value>192.168.0.5</AddressObj:Address\_Value>

</cybox:Properties>

</cybox:Object>

Both Objects are from two different Organizations, but contain the same Address\_Value of 192.168.0.5. They are effectively **exact duplicates** of each other but were **discovered by different Organizations** so should be linked in some way, **but we should still be able to retain the fact they were seen independently**.

Another related but slightly different scenario is where one Object is a superset of the other Object; where one Object contains only part of the other Object. In this document I’ll refer to this as **partial duplication**. As an example we have the two following CybOX Objects:

<cybox:Object id="myorg:Object-15be6630-c2df-4bf9-8750-3f45ca9e19cf">

<cybox:Properties xsi:type="AddressObj:AddressObjectType" category="ipv4-addr">

<AddressObj:Address\_Value>192.168.0.5</AddressObj:Address\_Value>

</cybox:Properties>

</cybox:Object>

<cybox:Object id="yourorg:Object-6e9d1bd0-e6ed-4ccd-bb8f-0ef0995b00a3">

<cybox:Properties xsi:type="AddressObj:AddressObjectType" category="ipv4-addr">

<AddressObj:Address\_Value>10.10.1.2##COMMA##192.168.0.5</AddressObj:Address\_Value>

</cybox:Properties>

</cybox:Object>

The ‘yourorg’ Object contains an IP address list, and the ‘myorg’ Object has just one IP address. Only one of the IPV4\_Addresses within the ‘myorg’ Object is a duplicate of the ‘yourorg’ Object. The question is – how would an implementer show this relationship, and is it something that we need to be able to reflect within STIX somehow?

## POSSIBLE ANSWER

There are a few ways of making de-duplication easier.

A potential solution to the **exact duplication** problem is to mandate that all STIX and CybOX Objects must have an Object ID generated from a combination of producer namespace and a hash of the contents, and to mandate this for all content production. In this way we can easily determine if objects have the same contents, as we would just need to compare the ‘hash’ part of the object ID. But this causes its own problem: it also breaks Incremental STIX Object Versioning (see “4. There are too many ways to update an Object (Versioning)” below). This is not too much of a problem, as we have recommended removing the Incremental Update mechanism, and **only** using the Major Update mechanism for changes to existing objects (which is basically a full reissue of a new Object with an explicit relationship with the old Object).

But detecting the same hashes in both Object’s IDs, we can then issue a Relationship Object containing a ‘duplicate\_of’ relationship (or something similar), to explicitly note that the two objects refer to the same thing e.g :

Duplicate\_of

yourorg Object

myorg Object

Detecting partial duplication is trickier. The use of the shorthand ‘list’ property of Observables (often used for IP address/domain name lists) causes issues in this respect. This effectively creates a situation where one object with a single property matches one item in a list within the property of the other object. It’s a 1:N problem.

One way to rectify this is to deprecate the list shorthand, and force each description of a single item to require its own Object. In this way objects that currently have a list of 5 IPv4Addresses within them would in the future be generated as 5 objects, each with its own Object. This has implications of storage, handling, bandwidth etc. Using this method would remove the partial duplication problem (at least at the CybOX Object level).

Another way to rectify this is for each implementer to relate them within their system, independently of STIX. Each implementer would separate out the list of IP addresses into individual objects within their solution (outside of STIX), and would relate them in a way like this:

myorg Object

192.168.0.5

yourorg Object

10.10.1.2

You can see from this diagram that the yourorg object has multiple objects within it (i.e. a list) and that the implementer has pulled those out into individual items to track. This method has its own problems, as there is no way for a discovered relationship to be shared.

Another way to deal with this is for the creation of a ‘partial duplicate’ or ‘contains’ style relationship type, and to just associate the two objects together with a relationship object:

Partial\_duplicate\_of

yourorg Object

myorg Object

# There are too many ways to update an Object(Versioning)

## PROBLEM

In the current version of STIX, an object is identified by a combination of its Object ID, and the Timestamp. One can think of the Object ID + Timestamp representing the composite primary key.

There are two ways to issue an updated STIX Object in the current version of STIX. (information here: <http://stixproject.github.io/documentation/concepts/versioning/>)

An Incremental Update only updates the Timestamp. This indicates there is a slight change to the underlying object, as it retains the original Object ID; the later timestamp indicates to the receiving consumer that the new object should overwrite the old one. It is an implicit update.

A Major Update creates a completely new STIX Object, with a modified Object ID + Timestamp. This indicates there is a major change to the underlying object, one large enough to cause a completely new object to be issued. The new STIX Object includes an explicit relationship to the original Object ID + Timestamp, with a type of 'Supersedes' to indicate to the receiving consumer that the new object supersedes the old one. It is an explicit update.

A large problem is that there is no definition (or agreement) as to what update mechanism should be used. The STIX guidance states:

*"Current suggested practices suggest using an incremental update whenever you're making very minor changes to a construct that don't change its inherent meaning. Adding an alias to a threat actor, for example, would be an incremental update. Additionally, incremental updates can be used within an organization while it is developing a more final version of the construct in order to avoid churn on IDs. Major updates, on the other hand, are suggested for anything that changes the inherent meaning of a construct or changes of content between organizations. Changing a TTP from "phishing" to "spear phishing", for example, would be a major update because even though phishing and spear phishing are similar the inherent meaning of the construct changed."*

Another problem is the recent discussion suggesting using hashes to generate the UUIDs for objects. The use of an Object ID + Timestamp as a composite key does not work with Incremental Updates. If a producer made a mistake and had to reissue the Indicator with a modification, that would result in a completely different Object ID if we were using a hashing function to generate the IDs.

## POTENTIAL ANSWER

This could be fixed by enforcing all updates to be new objects (with new Object IDs) with explicitly defined relationships to the old object IDs.

This would mean that:

* A STIX object only ever gets 'issued' once.
* That STIX object cannot be updated.
* An object that supersedes the original object must have a new Object ID, and a top-level relationship object describing that the new object supersedes the old object.
* The Object ID can be generated from a hash of the STIX Object contents

# Observable Patterns and Observable Instances differences aren’t easily discerned

## PROBLEM

Observables, Observable Patterns and Observable Instances aren’t easy for new Users to understand and discern the differences between. It took me months to realize that that Observables were actually made up of CybOX Observable Instances and CybOX Observable Patterns.

The difference between the CybOX Observable Instances and CybOX Observable Patterns and the rules to tell them apart are not obvious enough.

This leads on to the fact that Indicator Observables are often used to describe Observable Instances, when they really should be storing the things we are looking for - Observable Patterns. Observable Instances should instead be described separately within the STIX Observables construct and then referenced back as Sightings.

## POTENTIAL ANSWER

The name of Observable Instances should be changed to become STIX ‘Observations’. This would provide the following hierarchy:

**CybOX Observable -> STIX Observation**.

STIX Observations should be restricted from used within the STIX Indicator object.

The name of Observable Patterns should be changed to become STIX ‘Patterns’. STIX Patterns should only be allowed to live within the STIX Indicator Object, describing what one would need to look for in order for the Indicator to trigger. This would provide the following hierarchy:

**CybOX Observable -> STIX Pattern -> STIX Indicator**.

This will help greatly with Sightings (in conjunction with the top-level relationship object), as a Sighting now becomes as easy as sending a new Sighting object and a relationship object back to the producer of the Indicator. This will allow Indicator producers to get independent feedback from third-parties with Sightings of that Indicator. This will potentially help producers refine their Indicators to make them reliable.

# Difficult to ask a request and get a response to that question

## PROBLEM

There is no real mechanism within STIX for a consumer of STIX data to ask a question from the rest of the threat sharing community that they are part of. This functionality is required if we are going to get good multi-directional threat intelligence sharing happening.

**Scenario**

*A threat sharing community member has detected an IP address while doing some local network hunting that seems to be malicious, but they are unsure if it actually is. STIX needs to be able to allow the community member to send out a 'does anyone have information they can share about this' STIX request out to the entire community, and allow any other community member to reply to the community member. The replies may be shared with the entire community, or may be sent directly to the requester.*

This is different from the normal 'broadcast' style STIX message, where the message is just sent to all parties and no replies are expected. With STIX request/response there is a direct question/answer relationship required.

Please note this request/response is also different to TAXII Query, as the question is being asked to all members of the channel, rather than just the single TAXII server you are locally connecting to (which is IMHO more where TAXII Query fits in).

## POTENTIAL ANSWER

Creating a STIX Request Package and a STIX Response Package seems to be a good answer to this problem.

As I see it, a sender would have two types of questions they would want answered:

* I have a 'thing' (e.g. IP address), and I to find more objects that are related to this thing so I understand it better (crowdsourced responses)
* I have a particular STIX object and I would like to get the latest version of that object from the producer (particular object responses)

For the crowdsourced responses option, the STIX Request Package would contain a list of related STIX objects that the sender would like more information about. The STIX Request Package could contain something as small as a single IP address, or could contain a large slice of related data e.g. a list of 5200 Observables, Indicators, TTPs, Campaigns and Threat Actors. The STIX Request package would be sent to all recipients in the Threat Sharing Group, and any/all of the Threat Sharing Group members would be able to respond via a STIX Response package. STIX Responses from Threat Sharing Group members would be able to be sent to all Threat Sharing Group members (group reply) or sent directly back to the original STIX Request package author as a direct response (private reply). The STI Responses need to be able to say 'Yes, we've seen it, and we've included some objects that are related to it', or 'No, we've not seen it'.

For the particular object responses option, the STIX Request Package would contain a list of STIX identifiers that the sender would like more information about. The STIX Request Package would be sent directly to the producer of the object being queried. **This relies on the fact that STIX IDs include the producers namespace, that the namespace includes the domain name of the Producer, and that the producer has the relevant TAXII auto-discovery functionality enabled in their setup**. The producer would then look at the STIX Request Package author to determine if the producer wishes that information to be shared, and also check if the STIX Request Package author has the correct permissions to have access to that data. If they do then the data (or subset of the data) should be returned. This sort of STIX Request Package would always be sent back original STIX Request package author as a direct response (private reply).

Having both these features would enable more question and answers to be asked across threat sharing groups, meaning that Threat Analysts and Incident Responders would have the ability to find out more about their own particular use cases - hopefully improving the speed and effectiveness of Incident Response.

# Sightings are difficult to send out independently

## PROBLEM

In the current version of STIX Sightings are defined as Sightings of an Indicator. They are tied very closely to the Indicator, and are most often embedded within the Indicator. This close tie creates some problems which restrict the usefulness of Sightings:

* In order to add a Sighting to an Indicator the producer of the original Indicator would need to publish a new version of that Indicator with the new Sightings added.
* If a third-party wants to inform the producer that they have a Sighting related to the original Indicator there is no way to do so. The third-party will need to send the original producer a new Indicator with the similar details in it, but under the third-parties namespace, and then attach the Sightings they have to their own Indicator object. The producer of the original Indicator would need to somehow recognize that the Indicator is the similar to the same one that they sent out earlier, and would need to resend a new updated original Indicator with the third-parties Sightings attached.

## POTENTIAL ANSWER

This proposal requires the changes suggested within section 5 – “Observables, Observable Patterns and Observable Instances differences aren’t easily discerned”. In that proposal the name of Observable Instances (STIX Observables) would be changed to become STIX ‘Observations’. STIX Observations would be restricted from use within the STIX Indicator object.

Observable Patterns would be re-labeled STIX ‘Patterns’. STIX Patterns would only be allowed to live within the STIX Indicator Object, restricting them to describing what one would need to look for in order for the Indicator to trigger.

This separation of function would make the role of the Indicator and Sighting easier to understand for new users of STIX; The Indicator contains ‘things you should look for’, and the Observation contains ‘things you’ve seen’.

**Proposal**

* Rename Observable Instances to STIX Observations
* Rename Observable Patterns to STIX Patterns
* The new top-level Relationship Object (mentioned elsewhere in this document) is used to relate that fact a Sighting was detected. The relationship would link the Observation with the Indicator with a relationship type of ‘Sighting\_of’ or something similar.
* Observable patterns aren’t used within an Observation object, and instead live within an Indicator.
* The Indicator object is also restricted to only containing STIX Patterns, and loses the ability to contain Observations.
* Depreciate embedded indicator sightings.

This effectively creates a nice separation between 'things we need to look for' (Indicators+ Patterns) and 'things we have found' (Observations).

# Cannot generate a relationship between two third-party objects

## PROBLEM

Within STIX currently, you can only establish a relationship between a STIX object that you send, and one or more third-party STIX objects. You cannot assert a relationship unless you are also providing one of the objects. This limitation hinders third-parties who specialize in analysis and discovery of threat intelligence. We need to be able to provide them with a mechanism for saying 'we think these three objects from Org A are the same as these three from Org B and are part of Campaign FS-ISAC-X' even if those objects were created by other entities.

## POTENTIAL ANSWER

A top level relationship object will enable this independence. It will allow a third-party to assert a relationship between two data objects that it doesn't 'own'. This relaxing of the restriction will allow for greater 'crowdsourcing' of opinions as the options will be able to be given even if the producer doesn't have sightings of their own to contribute. It will enable people with a collection of third-party information to help others establish relationships with objects outside their namespace.

In addition, if we allow relationship objects to be sent independently of STIX data nodes (the STIX objects that describe entities) then Organizations can still match up the fact that there are relationships even if they are not allowed to actually receive the relevant STIX data node. As an example, Org A sends out a new Incident B object to a Threat Sharing group. Gov X realises this Incident is related to Campaign Y they are tracking, but they don't want to send out the details. So they instead just send out a relationship object that ties Incident B to Campaign Y, but they don't send Campaign Y object. Later on Org B sends out details of Incident C along with a relationship object that ties Incident C to Campaign Y, and Org A sees that. Org A now is able to relate Incident C with Incident B via Campaign Y even though they don't have the Campaign Y details. The relationship still provides value.

I suggest the following rules regarding the relationship object:

* Make relationships a top-level object. All other STIX objects are considered STIX data nodes
* Force relationships to exist ONLY if there is a relationship object linking two STIX data nodes together
* Multiple relationships are shown with multiple single relationship objects
* Not allow the ##COMMA## notation any longer, but replace with multiple single relationship objects.
* Relationships are allowed to be sent independently of the STIX data nodes they refer to.

The reason that I am suggesting 'single' relationship objects is to allow third-parties to [+1] or [-1] each individual relationship. This would allow consumers to evaluate who to trust – if a relationship had only 1 [+1]’s and 45 [-1]’s then you know it’s probably a low confidence relationship and one that shouldn’t be trusted as much as others. See later in this document.

# Too many ways to do things

## PROBLEM

There are too many ways to represent things in STIX, and this has a knock on effect when it comes time to implement code. We really need to find a way of simplifying things, so that we have one mandated way to do it that all parties are required to support, and that is the recommended way of doing things. The main way needs to support the 80% of situations. We also need to support niche customized extensions to STIX to still allow the other 20% to implement what they need, but the main functionality needs to be well-defined and simple.

As an example of the problem, below is a list of the currently ways a relationship between two IP addresses can be described in STIX. Any STIX compatible solution needs to be able to handle ingesting data in all of these forms. This creates extra work and difficulty for implementers.

|  |
| --- |
| 12 ways to express context between 2 IP addresses |
| 1 | Indicator, with two inline IPv4 AddressObjects |
| 2 | Indicator, with two referenced IPv4 AddressObjects |
| 3 | Indicator, with one inline IPv4 AddressObject using comma notation (127.0.0.1##comma##127.0.0.2) |
| 4 | Indicator, with one referenced IPv4 AddressObject using comma notation (127.0.0.1##comma##127.0.0.2) |
| 5 | A composite indicator including a single indicator, with two inline IPv4 AddressObjects |
| 6 | A composite indicator including a single indicator, with two referenced IPv4 AddressObjects |
| 7 | A composite indicator including a single indicator, with one referenced IPv4 AddressObject using comma notation (127.0.0.1##comma##127.0.0.2) |
| 8 | A composite indicator including a single indicator, with one inline IPv4 AddressObject using comma notation (127.0.0.1##comma##127.0.0.2) |
| 9 | A composite indicator with two indicators. Each indicator has a single inline IPv4 AddressObjects |
| 10 | A composite indicator with two indicators. Each indicator has a single referenced IPv4 AddressObjects |
| 11 | Two AddressObjects, no indicators, and "These IP addresses are malicious" placed in the Title field of the STIX\_Header (implicit relationship) |
| 12 | One AddressObject using comma notation (127.0.0.1##comma##127.0.0.2), no indicators, and "These IP addresses are malicious" placed in the Title field of the STIX\_Header |

## POTENTIAL ANSWER

This is a general issue with STIX, CybOX and TAXII. There is an abundance of flexibility built into the protocol. This flexibility in turn causes its own problems, as it introduces complexity and makes it much harder for implementers to actual code for the multitude of different scenarios that are required to capture the variations allowed in the standard.

We need to decide what the high-level goals of the STIX v2.0 development are; what are we actually trying to achieve with STIX v2.0. I believe the following list is a good start:

Goals (developed from the TAXII group goals):

* **Simplicity**
	+ Easy to implement and understand
	+ One way of doing things where possible
* **Reduce optionality**
	+ Support Customization in a simple, standardized way
	+ Don’t allow customization everywhere, only where it is likely to be used
* **Standardization**
	+ Do things in the same way across STIX and CybOX
	+ Reuse similar structures in similar yet distinct parts of the model
* **Modularity**
	+ Provide building blocks that can be reused elsewhere
	+ Ensure tight cohesion, and low coupling of those building blocks
* **Flexibility**
	+ Use modularity to provide flexibility
	+ Allow relationships to exist between any objects
* **Better Analysis**
	+ Easier to graph relationships
	+ Easier to track changes over time
	+ Easier to put together timelines
* **Minimize resource usage**
	+ Reduce size where possible
	+ Only transmit what is necessary
* **Target the 80%**
	+ Concentrate on the common scenarios (Use Cases)
	+ Work on the other 20% in subsequent releases

For the specific scenario described above, there are a few issues at play:

* The ability to either Inline or Reference content
* The ability to either create Composite Indicators or a list of items within an object to reflect grouping
* The ability for either an implicit or explicit relationship

All of these 12 ways are trying to do a fairly simple thing - show a relationship between two objects. I believe that the problems can be fixed here and in other parts of the STIX data model by adding a few changes/rules (as described above in the previous section):

* Make relationships a top-level object. All other STIX objects are considered STIX data nodes
* Force relationships to exist ONLY explicitly i.e. if there is a relationship object linking two STIX data nodes together
* Multiple relationships are shown with multiple single relationship objects to allow for easier
* Deprecate the ##COMMA## notation, and replace with multiple single relationship objects.
* Relationships are allowed to be sent independently of the STIX data nodes they refer to.

The reason that I am suggesting 'single' relationship objects is to allow third-parties to [+1/-1] each individual relationship, and to maintain individual objects so that the timeline of the order that things happened can be traced. This will help when analysis is performed targeting a particular sequence of events.

# Indicators/Observable, Composition/Object hierarchy is complicated

## PROBLEM

When complicated Observable Patterns are received it is difficult for implementers to store, extract and use them. We need a way of simplifying the patterns themselves. Observable instances are fine, as they are always related to each other - therefore being an AND type relationship - which will make the creation of the new Sighting Object reasonably straightforward. Observable Patterns on the other hand are potentially looking for slight variations of particular observables, meaning that the patterns can be quite complicated to describe these variations.

As an example…the following Observable pattern indicates that this is a bad email:

* The email is from hackedman@example.com OR hackedwoman@example.com
* AND the email has the subject 'free tickets to ManU' OR 'your parking ticket'
* AND the email contains a PDF document attachment
* AND NOT the email also contains the word 'SENSOR DETECTED'

This can be pretty complicated to describe, especially with the multiple different ways this can be described – Indicator Composition, Observable Composition or Related Objects.

## POTENTIAL ANSWER

There must be a way of ‘flattening’ the Observable Composition/Indicator Composition hierarchy to still reflect the relationships but without a huge multi-level object within object within object structure.

It could be as easy as using a Boolean expression as a string, using Object IDs, Boolean operators and nested parenthesis. But something that generates a smaller output size would be beneficial, as the observables and indicators are likely to be the most numerous, and would benefit most from a size reduction.

One possible way of achieving this comes as a by-product of enforcing reference only. We could have a single Indicator Boolean string that references the ID’s of the STIX Pattern (Observable Pattern) that are used in the Indicator. This would work as follows (in JSON Notation):

{“stix:Indicators” : [

{

“id”:"example:indicator-8cf9236f-1b96-493d-98be-0c1c1e8b62d7",

“timestamp”:"2014-10-31T15:52:13.127931+00:00",

“version”:"2.1.1",

“title”:”Malicious E-mail”,

“type”:"Malicious E-mail”,

“Condition”:"example:Observable-437f0c20-ab26-4400-9f6a-fc395da3ddd9 AND (example:Observable-437f0c20-ab26-4400-9f6a-fc395da3ddd9 OR example:Observable-437f0c20-ab26-4400-9f6a-fc395da3ddd9)",

“confidence”:”High”

}

]}

Please note the STIX Patterns (Observable Patterns) would be defined in their top-level Object array, and the Indicator Condition would be used rather than the top level Relationship instance. The reason for this is that the top-level Relationship object would not have the ability to describe the Condition properly.

It has been suggested that the current structure is actually very good for implementers, as it is reasonably straight-forward to parse that as an Abstract Syntax Tree. Ultimately the producer is trying to extract the Boolean logic in whatever form it takes, and parsing a Boolean string that contains or parsing a linked hierarchy of objects. Flattening these layers would still enable the AST to be derived from the Boolean condition.

In addition, we should look at whether Indicator\_Composition and Related\_Objects are actually required in STIX v2.0. Anecdotally it seems most people are using Observable\_Compositions if they need to compose a more complicated Boolean pattern match. This may indicate that Indicator\_Composition and Related\_Objects are not required in STIX v2.0.

Section 23 - “Which to use? Indicator Composition, Observable Composition, or referenced Object?” discusses this further.

# Embedded and Referencing Objects

## PROBLEM

If you want to show a relationship between two objects within STIX at present, you can either create a top-level object and reference that object from another object (Reference), or you can embed the object directly within the other object (Embed).

This embed vs reference decision complicates matters for implementers. Rather than just supporting one way of doing things, implementers need to support both ways of tracking the objects, complicating things at the deserialization step.

## POTENTIAL ANSWER

We can rectify this issue by mandating that all relationships between objects that **will likely be related to multiple other objects** must be made using the relationship object (discussed elsewhere in this document) – never embedded. Any Object that is likely to be reused again must be created independently, and have its own referenceable Object ID. The object will then be referenced from the other Object.

If Objects will only ever have a 1:1 relationship with one other Object, then they should actually be flattened into the containing Object to become part of the containing Object itself.

We envisage all relationships between objects being described using a separate independent Relationship object (see section 8 - Cannot generate a relationship between two third-party objects), all except for the Indicator to Pattern (Observable Pattern) Relationship, where the Boolean Condition described within the Indicator would describe the pattern that ‘triggers’ the Indicator.

# Difficult for third parties to show agreement or disagreement

## PROBLEM

At present it is nearly impossible for third parties to indicate their agreement or disagreement with another Organizations assertion. The ability to indicate agreement is critical for the following useful functions:

* The ability to [+1] a relationship to confirm your agreement.
* The ability to work out which source to trust
* The ability to recognize that an Organization believes something that is not commonly thought
* The ability to ignore a particular information source as a source of disinformation.

There is no current way for a third party producer to disagree with content published by another party. For example, if Org A pushed out content that described 8.8.8.8 (Google DNS) as being malicious, we need a way for the community to 'push back' this assertion. A way for the community to say I disagree with this assertion.

In a similar vein, we need the ability for Organizations to agree with assertions that other Organizations make. If Org A sends out content that says www.baaadsite.com is malicious and is part of the Dyre malware family, and Orgs B and C agree, then we need to give them a way for Orgs B and C to share the fact that they [+1] that assertion.

At present the best Orgs B and C can do is to send out their own TTP/Campaign object with references to a combination of their own Indicators with their similar evidence, and references to the Org A Indicators. They then need to hope that other members of the Threat Intelligence sharing community work out the relationship to the Org A TTP/Campaigns themselves.

At present Org B and Org C cannot disagree directly with Org A through STIX, but instead can only produce their own assertion of what happened.

## POTENTIAL ANSWER

One way this could be fixed is with the creation of a top-level Opinion Object. This object would provide the ability to **explicitly** agree/disagree with another object. The opinion object would be allowed to refer to STIX data objects (e.g. Indicator, Observable, TTP, etc) or directly to Relationship objects, allowing producers to either agree/disagree with the data or the relationships between the data.

The Opinion Object would be small, with basically the following psuedo-structure:

opinion\_object {

 id: QName # this id of this opinion

 timestamp: # just so that this can be updated at a later date ()

 version: # the version of this opinion object (e.g. '1')

 object\_ref: QName, # the id of the referred object.

 agreement: ['strongly agree', 'agree', 'undecided', 'disagree', 'strongly disagree', 'unknown'],

 description: StructureTextString,

 source: InformationSourceType

}

## ANOTHER POTENTIAL ANSWER

This could also be fixed by just defining how we use a top-level Relationship object, such that sending a +1 actually results in a producer sending their own relationship object (with an ID in their namespace) that has links between the IDs of the objects they agree/disagree with. Rather than explcitly disagreeing with another Orgs assertion of a relationship, this method would require the consumer to recognize that there are 2 relationships saying there is a relationship and 5 saying there is not a relationship. Then it is up to the recipient to decide who to trust (as it should be).

As an example, imagine that Org A sends out Indicator A, TTP A, and a relationship object Rel A that links the two together to show they are related. Org B disagrees, and sends out a Rel B object which still refers to Indicator A and TTP A, but which asserts there is not a relationship.

The disagreement is not explicit. It is up to the consumer to determine the fact the competing relationship objects are referring to the same Objects.

# Obfuscation of Producer **Content** is Difficult

## PROBLEM

Sometimes some Organizations want to tell other Organizations to monitor for some Indicators, without having to tell them why. This may be for 'National Security' reasons, commercial confidence reason, or even just licensing requirements. We need a way for people to understand how objects are related WITHOUT them having access to the related data.

## POTENTIAL ANSWER

It should be perfectly acceptable for only relationships to be shared, and no actual data objects to be shared (although it wouldn't be that useful). It is important for the details in to be obfuscated if the producer wishes them to be. With the current version of STIX there is no way for a producer to acknowledge that there are relationships to a STIX object and yet restrict access to the information in the object itself. By allowing relationships to be sent independently of the STIX data objects that it refers to, we allow the consumers to link relationships to each other without needing to understand what information is actually within the STIX data objects.

# Obfuscation of Producer **Identity** is Difficult

## PROBLEM

Sometimes some Organizations want to tell other Organizations some information, without them knowing it came from them. This may be for 'National Security' reasons, commercial confidence reason, or even just licensing requirements. We need a way for secretive organizations to provide data without their identity being disclosed, yet we also need their identifier to relate specifically to them so that they can receive Sightings as feedback, and so that others in the Community can respond to the information they provide.

## POTENTIAL ANSWER

We can take some lessons from the networking world. When an Organization wants to keep it's internal network ranges secret, it can use Network Address Translation (NAT) to 'hide' this information behind a NAT proxy, which will translate the real IP address into a fake IP address as the traffic passes through it.

Similarly, we can use the concept of a STIX proxy, where content generated in the namespace of a secretive organization can be hidden behind a STIX proxy, and where all content eminating out of the external side of the STIX proxy is within the namespace of the Organization operating the STIX proxy.

As an example, if Gov Dept X wants to send out a list of Indicators X to all OASIS members, yet doesn't want anyone to know where it came from, it could use a third-party STIX proxying service (lets call it SProxY) to hide its content behind. Gov Dept X could sent SProxY it's Indicator X, with the request that SPRoxY sends out the content on it's behalf. SProxY will then distribute the content within the SProxY namespace, making it appear to everyone who views the content that it came directly from SProxY. SProxY has effectively NATed the communication; everyone thinks it is SProxY's information but only SProxY knows the truth.

This proxy style obfuscation has the added benefit of allowing bi-directional communcation, meaning that if anyone else releases content that adds to or enhances the content using the SProxY identifier, SProxY is able to see that, translate the SProxY identifier back to the original GovDept X identifier, and give the Gov Dept X that information.

It means that Gov Dept X is able to fully participate in trust groups without the participants of the trust groups knowing the original source of the information.

# Difficult to ask for a referred object

## PROBLEM

If all objects become top-level objects and if all objects are referenced (not embedded), then we cannot assume that all objects will be actually available or shared at the same time that their relationships are shared. It is quite possible that a relationship is shared but the STIX data object it refers to is not.

## POTENTIAL ANSWER

This will be fixed if we allow STIX Request and Response packages. See the commentary discussed in the 'Difficult to ask a request and get a response to that question' section.

# Victim Targeting is embedded within a TTP

## PROBLEM

There are two types of information related to the Victim that are useful for defenders to have:

* Specific information about a particular victim and what they lost (specific victim info)
* General categorized information about what types of victim the attacker targets (general victim info)

Details about the Victim Organization are currently embedded within the TTP object.

<http://stixproject.github.io/data-model/1.2/ttp/VictimTargetingType/>

Having this Victim Targeting information embedded within the TTP object restricts users from being able to document the Victim Targeting independently of the TTP object. This means that information about a Victim being targeted cannot be shared unless a TTP object is generated. If the Victim doesn’t know any details about how they were hacked they would need to release a TTP empty except for their victim details.

The TTP appears to be focused on the general victim info.

## POTENTIAL ANSWER

### Specific Victim Info

By pulling out the Victim Targeting into its own object we enable the information about the Victim to be shared without knowledge of how they were hacked/affected. It means that someone can effectively notify they were targeted, and can then fill out and relate the TTP when they learn more. The producer can create a TTP later and relate them together when they learn how the attack took place.

APT Threat Actors often target Organizations based on the type of work they do, their position in the Defense Industrial base, Intellectual Property that they have, customers they may have, infrastructure they run or any other feature about them that is useful to the Threat Actor.

# No STIX-wide way to handle aliases

## PROBLEM

Different Organizations call different entities different things. It is often difficult to understand that two Objects with different names are actually the same thing. We currently don't have a way to track aliases that will apply to all STIX data objects. At present you can track Aliases in only a few objects – for example the Threat Actor Object via Related Identities. We need to be able to do this for all the Objects, such as TTPs, Campaigns and the like.

In many ways this is similar to the deduplication problem.

## POTENTIAL ANSWER

There are a few ways this could be achieved:

* We could provide an Alias object, and have a relationship type of ‘also\_known\_as’ to allow certain specific objects to be known with other identifiers.
* We could use the ‘Investigation/Tag’ object as a label/tag facilitator, and use it to ‘group’ the objects that use the alias together.
* Another option is to just deal with it using the de-duplication processes mentioned earlier, and to directly relate the two Objects together as per section 3 – “Deduplication is difficult”.

It probably makes sense to use the last option as it reuses the relationship object to its fullest extent, and reduces the amount of extra Objects STIX needs to support..

# Interoperability is difficult if custom vocabularies are allowed

## PROBLEM

Default Vocabularies aren't expressive enough to accurately describe what producers want. For this reason we expect that producers will want the ability to use customer vocabularies. The problems with using custom vocabularies is that it becomes difficult for receiving implementations to processing them.

## POTENTIAL ANSWER

In order to improve interoperability of solutions from different vendors, we need a way of ensuring a standard set of vocabularies. This can be achieved in two different ways.

* Only have a single, official, centrally controlled default vocabulary. Make sure that it is extremely easy to request additional vocabulary entries, and make sure that updates are automatically distributed to vendors. The vocabularies MUST be able to be distributed independently of updates to STIX, CybOX and TAXII.
* Have a high-level official, centrally controlled default vocabulary, and allow producers to provide their own externally referenceable custom vocabulary (housed at a reachable URI) if they wish. Doing things this way ensures that there is the high-level 'fallback' vocabulary available if the consumer doesn't have the ability to contact the publically reachable custom vocabulary URI.

# Difficult to group 'possibly' related things during an investigation

## PROBLEM

There is no easy temporary way to temporarily relate **possibly** related items together. When one is conducting an investigation into a series of suspicious events prompted by your Organization’s monitoring processes, we often want to tag/relate these events together, without actually creating an official ‘Incident’ (as we’re not sure anything has actually happened yet – it could be a false positive).

It is currently possible to put that information inside an Incident object, but I view the Incident Object as somewhere where one would put the information when it is **confirmed** there is a problem. I believe we need a separate way of ‘tagging’ and ‘grouping’ potentially related items together in more of a throwaway fashion.

This may have a lot in common with section 20 “Cannot suggest hypotheses to a community through STIX”.

## POTENTIAL ANSWER

There are a couple of ways of doing this.

Firstly we could just use the top-level relationship object to link potentially related objects together in as described in section 20 “Cannot suggest hypotheses to a community through STIX”. There could be some kind of marking within the relationship that would allow the relationship to have a type of ‘hypothesis’ (or similar), and the Incident could have a status of ‘Under Investigation’. This would allow the Incident Object to become the ‘tag’ that relates all the other objects together.

The downside to this design would be that the grouping mechanism (the Incident in this case) will only apply to the things that are can reference the Incident, potentially leaving out other objects that don’t have a direct relationship within the current data model. If we decide to allow relationships to occur from any STIX object to any STIX object then this mechanism could work.

A second way to accomplish this would be to create a ‘tag/label’ Object. This would solely be a grouping mechanism, allowing one to link (using top-level relationships) from that tag object to any other STIX Objects. One could link any Objects to the tag object, and keep that relationship for as long as required. It could be beneficial for grouping potentially related items together, and would allow for related items to have multiple labels at the same time. A STIX incident and exploit target could all be related to a ‘TorrentLocker’ tag object, an ‘Investigation#352’ object and a ‘weirdly formatted spam’ object all at the same time.

# Cannot suggest hypotheses to a community through STIX

## PROBLEM

There are no certainties in Threat Intelligence Gathering and Analysis. Everything bit of information you receive should be treated as the author’s assertion of the truth – it is not the truth.

Threat Analysts are looking for patterns; looking for commonality; looking for statistical outliers. And when they find something unusual they have a need to track it in some way for future use and investigation. They are effectively looking at the collection of data that they have received from others and themselves over a multitude of mechanisms, and trying to make sense out of it. They slice it, dice it and try to form new relationships between the objects within it - new 'hypothetical' relationships that range from nearly impossible to purely speculative.

We need to support the both ability to share these more hypothetical relationship possibilities to help the threat analysts speculate, yet we need to allow the incident responders at the coalface to only care about the immediate threats and provide them with the ability to defend their Organization from attack.

At the moment we have the ability to say 'we assert that Object A and Object B are related with low confidence', but we don't have the ability to say 'if Object A was related to Object B then that would mean Objects C, D and E are also related'. If we provided the ability to send out hypothesis and get agreements and disagreements with the hypothesis sent back to the originating Threat Analyst (à la indicator sightings) then they would enable the Threat Analysts to crowdsource 'what-if' scenarios amongst themselves, leading to potentially faster conclusions.

## POTENTIAL ANSWER

This could be handled within the relationship object, by somehow acknowledging the hypothetical relationships are exactly that. Providing the mechanism for separating hypothetical relationships with real ‘production-level’ relationships will allow people to use only the production-level relationships in their security tools, yet still keep track of the hypothetical relationships and participate in community speculation.

This section goes hand-in-hand with the Investigation Object idea (section 18: Difficult to group 'possibly' related things during an investigation”).

# Relationships are constrained to limited Objects within STIX

## PROBLEM

There is a limited number of relationships that are ‘valid’ within STIX. This can limit the ability of objects to directly show their association with another. As an example an Indicator cannot currently be directly related to a Threat Actor, but instead MUST go through either a TTP object, or a Campaign. Similarly neither a Campaign nor Incident can be directly associated with an ExploitTarget. Yet there may be times that this is exactly what a user wants to reflect.

A supplied example as to why this is not helpful.

*“Say badguy1 uses CVE-10. Right now, I have to link that via a ttp. There are two possible scenarios here. The first is that I might not know the ttp being used to exploit CVE-10 The problem with that is obvious, no link can be created currently. The second case is if I do know the ttp. This represents a linkage that can be created, but at a loss of clarity/specificity. So let’s say badguy1 uses CVE-10 via spearphishing. So I link badguy1 to spearphishing and link spearphishing to CVE-10. BUT, I also have 20 other CVEs linked to spearphishing, and 100 other threat actors linked to spearphishing. At that point I have no way to tell which bad guy uses spearphishing to exploit which CVE.”*

## POTENTIAL ANSWER

If we could remove the constraints around relationships that they follow the current STIX Model then we could allow any STIX top-level Object to be associated with any other STIX Object. This flexibility would make it easier for anything to relate to anything, and allow the model to reflect reality (or the producer’s assertion of reality) without constraint.

We would be effectively providing the basic building blocks that end-users would be able to use to construct the content that reflects their reality. We won’t be imposing our interpretation of how things are structured, but instead allowing users to model the relationships where they exist. Please note this still will require further research and planning, as we will need to add to the relationship vocabulary to provide an ‘agreed list’ of the types of relationships one can describe. This will take some time to fully populate.

# TTPs are **almost** mandatory

## PROBLEM

If you have an Indicator, and you wish to send that out in a manner that matches best practice, you are encouraged to use a TTP, even if that TTP does not add that much value. In addition one requires either a TTP or Incident in most cases to connect an Indicator to other things. As mentioned above in section 21, it may be worth investigating if this does actually need to be the case. It may be worth creating more flexibility in the relationships that are allowed within STIX.

## POTENTIAL ANSWER

This may be more of a tooling problem or ‘best practice’ recommendation problem than actually a problem with STIX.

Please see section “21. Relationships are constrained to limited Objects within STIX” above.

# Which to use? Indicator Composition, Observable Composition, or referenced Object?

## PROBLEM

There are multiple ways that Indicators and Objects can be composed/related together as part of an Indicator. If a producer has discovered an Indicator of badness that describes an email with an attachment, there are a few different ways of describing that:

|  |
| --- |
| Too many ways to compose Indicators, Observables and Objects together |
| 1 | A composite indicator including two indicators, with the first referencing the email Observable with a single email object, and the other Indicator referencing the attachment Observable with a single attachment object |
| 2 | A single indicator including an Observable Composition, with the first Observable containing a single email object, and the 2nd Observable containing a single attachment object |
| 3 | A single indicator including a single Observable containing two Objects – the first Object describing the email, and it containing a Related\_Object reference to the single attachment object. |

This is multiple levels of variability, and very confusing for new users of STIX. There must be a way of making it simpler – or even better restricting it to the ‘one way to do it’.

## POTENTIAL ANSWER

All three layers of variation may not be required. *Anecdotally* it seems most people are only really using Observable\_Compositions. This may indicate that Indicator\_Composition and Related\_Objects are not required in STIX v2.0.

**We should do a survey to see who is using what, and use that evidence as the basis for our future design.**

Section 24- “Are CybOX <Object> IDs used in STIX?” has some details on the use of Object ID’s as there have been some questions whether Cybox:Objects actually need IDs at all. This topic and that topic are closely related.

# Are CybOX <Object> IDs used in STIX?

## PROBLEM

There is some confusion as to whether cybox:Object ID’s are actually useful within STIX. Using the example below:

<cybox:Observables>

 <cybox:Observable id="example:Observable-e8ecf273-4f4d-46f2-834e-9381c30799d5">

 <cybox:Object id="example:Object-8108c0dc-bded-4b0c-b423-8b92ef1d6503">

 <cybox:Properties xsi:type="AddrObj:AddressObjectType" category="ipv4-addr">

 <AddrObj:Address\_Value>192.168.1.1</AddrObj:Address\_Value>

 </cybox:Properties>

 </cybox:Object>

 </cybox:Observable>

</cybox:Observables>

Most of the time the Observable is the one being referenced. The Object appears to be a layer that is there to add flexibility to a structure that is already potentially ‘overly’ flexible.

Implementers are required to record and track the cybox:Object IDs, even though they appear to be infrequently used.

## POTENTIAL ANSWER

Do we actually require both the CybOX Object and CybOX Observable layers? Can they combined together without too much loss of flexibility? Maybe something closer to the following structure:

<stix:Observation id="example:Observation-fe2c7b98-6e75-462b-b94f-e18024d4f4f0">

 <cybox:Object id="example:Object-8108c0dc-bded-4b0c-b423-8b92ef1d6503">

 <cybox:Properties xsi:type="AddrObj:AddressObjectType" category="ipv4-addr">

 <AddrObj:Address\_Value>192.168.1.1</AddrObj:Address\_Value>

 </cybox:Properties>

 </cybox:Object>

</stix:Observation>

**Observation** has an **Object** that has **Properties.**

This topic is closely related to section 23 – “Which to use? Indicator Composition, Observable Composition, or referenced Object?”, and should probably be discussed as part of that.

# Some Object names are confusing

## PROBLEM

Some of the Object names currently used within STIX and CybOX have certain connotations associated with them which color the way that those Objects are viewed, and therefore used. Some comments we’ve heard from people when discussing with them are that ‘but that’s what the Object is called’.

The objects that have been pointed out to us are:

* **Incidents**
Within Incident Response circles the SOC Analyst performs an Investigation, and then calls an Incident when he/she has confirmed that malicious activity is occurring. This contrasts with the STIX Incident, which was developed for use at all stages of the Incident Response lifecycle.
* **Test\_Mechanism**

Most people when told of the Test\_Mechanism idea say ‘oh like Signatures?’, which indicates that we’re probably using the wrong word. The complicating factor is that there are also OVAL and OpenIOC test mechanisms in there which have quite a different purpose to the rule focused snort and yara test mechanisms.

* **ExploitTarget**

Most people I’ve spoken to have no idea what this is, and have to have the concept explained to them. Maybe this Object is actually conflating vulnerability, weakness and misconfiguration together?

* **Observable Instances and Observable Patterns**

As mentioned in section 5 “Observable Patterns and Observable Instances differences aren’t easily discerned” many people find Observable Instances and Observable Patterns extremely confusing, and often interchange their use – especially in Indicators.

## POTENTIAL ANSWER

To make STIX more approachable, we should survey the community to find out if there are any other names that they find confusing, and attempt to come up with replacements that make more sense to the STIX and CybOX populace. Some suggestions for alternative names are listed below:

* **Incidents**
If we decide to create a new Investigation Object (see section 19) then this object can retain its current name. But if we do decide to keep the Incident Object an expand its functionality then its name should likely be changed to reflect that its scope covers the Investigation and Security Incident phases of the Incident Response process.
* **Test\_Mechanism**

I’ve only ever seen Snort rules used in a test mechanism. My personal preference would be to change the name of test mechanism to one of the following:

* + Rule
	+ Signature
* **ExploitTarget**

It could be worth separating the ExploitTarget information into 3 different sections:

* + Vulnerabilities
	+ Weaknesses
	+ Configurations

This would enable

* **Observable Instances**

As mentioned in section 5 “Observable Patterns and Observable Instances differences aren’t easily discerned” many people find Observable Instances and Observable Patterns extremely confusing, and often interchange their use – especially in Indicators.

Some suggested alternative names:

* + Observation
* **Observable Patterns**

As mentioned in section 5 “Observable Patterns and Observable Instances differences aren’t easily discerned” many people find Observable Instances and Observable Patterns extremely confusing, and often interchange their use – especially in Indicators.

Some suggested alternative names:

* + Criterion (preferred)
	+ Pattern
	+ Trigger
	+ Rule
	+ Match
	+ Test
	+ Parameter
	+ Check