STIX 2.1 - Cyber Observables Specification

Working Concepts

Document Table of Contents

Document Development Status

TODO - This section should be removed from the document prior to completion. It is included here to help visually track where things are at in the process.

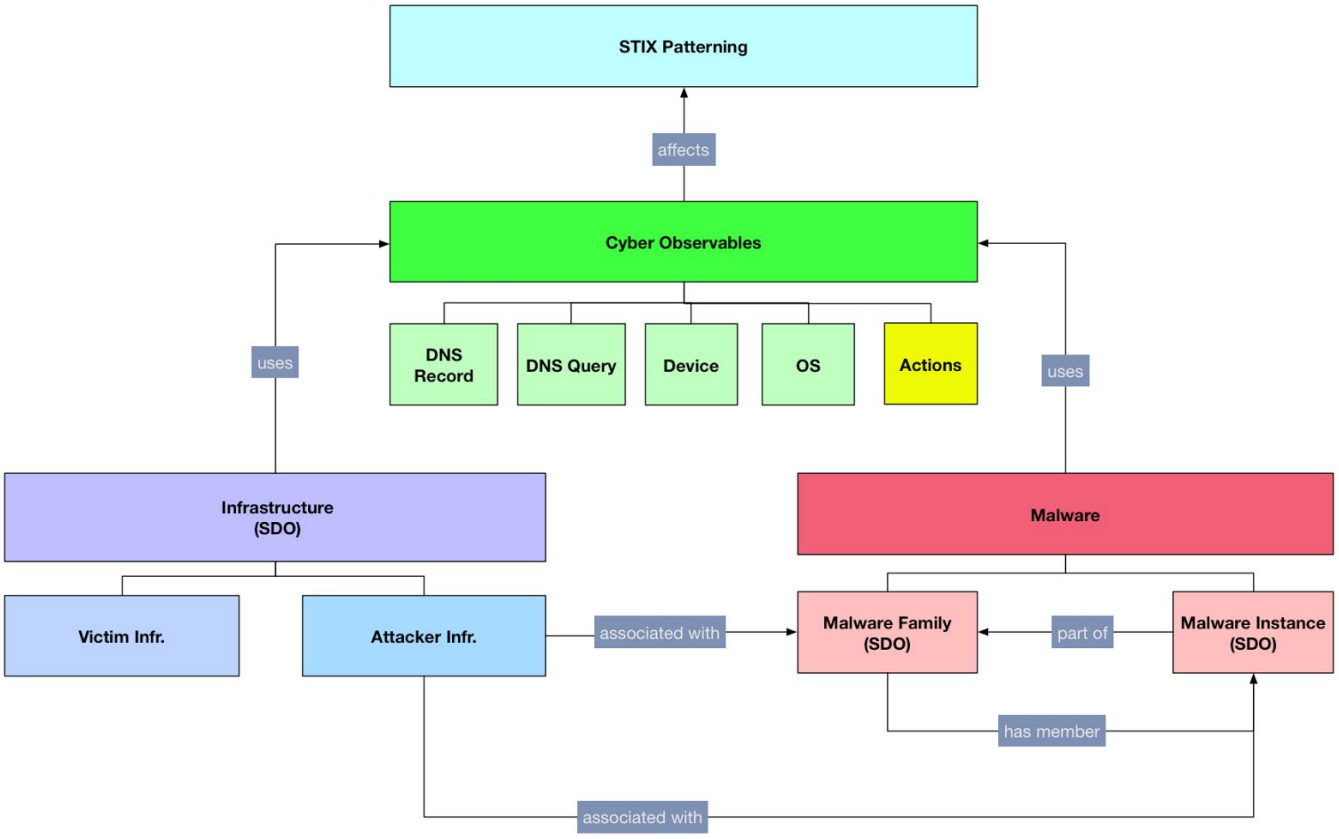
Each physical documents contains a table that defines 4 levels of development for each CTI topic. The first level is called Concept. Content coming in to one of the documents starts as a idea / concept. Once the community begins to work on the topic it will move to the Development phase. During this phase, the group will flesh out the design and come up with normative text. As the group comes to general consensus it will move to a Review phase. In order for a topic to move to its final Consensus phase, a formal motion will be made on the email list. Draft status doesn't mean that the text cannot change. Editorial changes can be made throughout the process without going back to earlier phases, however, if material changes are needed, the topic under review would move back to the Development phase and start again.

Concepts	Status
`action`	Concept
`Patterning Extensions` • `Patterning on STIX?` • `Back-references?` • `Variable substitution?` • `Functions?`	Concept
Objects	**Status**
`passive-dns`	Development
`device`	Development
`os`	Development
`webpage`	Concept
`dns-request`	Concept
`dns-response`	Concept

`whois`	Concept

1. 1. Cross-Cutting Changes

1.1. Malware & Infrastructure



1.1.1. Mission statement

- Develop forward-looking, properly scoped data models that we can leverage now and expand upon as needed in future releases.
- Don't try to do **EVERYTHING** in STIX 2.1.
- Don't reinvent any wheels - use existing data models/resources wherever applicable.
    - OSQuery
    - Splunk CIM

1.1.2. Use Cases

- Infrastructure
    - Characterizing malicious infrastructure used by adversaries

        - C2
            - Malicious web hosting/compromised web servers
        - Botnets
            - Malware hosting
    - Characterizing victim infrastructure targeted by adversaries
    - Characterizing victim infrastructure that has been actively exploited
        - i.e., in Incident
    - Characterizing victim infrastructure that is susceptible to exploitation
    - Patterning on vulnerable devices for detection/remediation
    - Forensic characterization of compromised/affected devices
    - Virtualized infrastructure characterization
- Malware
    - Malware instance characterization
        - Sample identification
        - Malware instance naming/aliases
        - Capture of common (~90%) analysis results
            - Static analysis results
                - Strings
                - Certificates
                - Import hashes (imphash)
                - Signatures (opcodes)
            - AV classification results
            - Dynamic analysis/sandbox results
                - Actions (?)
            - Network elements
                - C2 callouts (domains, IP addresses, etc.)
    - Malware family characterization
        - Family naming/aliases
        - Capture of properties common to all malware instances in a family

1.1.3. Potential Objects

- STIX SDOs
    - Infrastructure
    - Malware Family
    - Malware Instance
- Cyber Observable Objects
    - Device
    - Operating System
    - DNS
        - DNS Query
        - DNS Record/Passive DNS

1.1.4. Open Questions

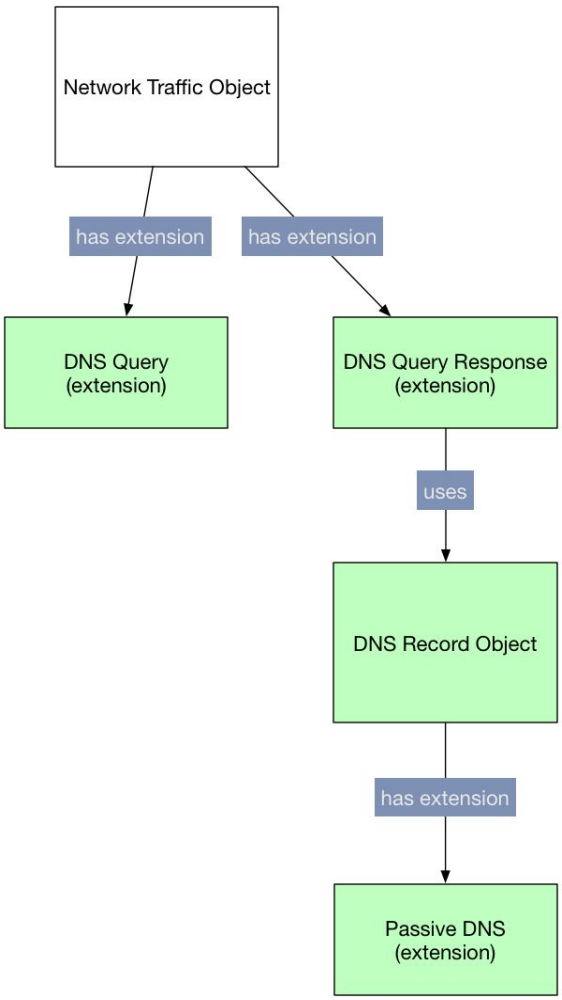
- Scoping - is our scope for malware and infrastructure too narrow or too broad?
- How do we create a forward looking data model?
    - There needs to be an outreach component - we must get in touch/get feedback from the right participants.
- What other Cyber Observable Objects do we need?
- Malware-specific
    - Memory resident/advanced malware - how do we characterize it? Should we even try?
    - What types of malware analysis results should we capture?

        - AV/classification results
        - Dynamic analysis/sandbox results
            - Requires Actions
        - Static analysis results
- Infrastructure-specific
    - Is one object too abstract? As with Malware, should we try to break this concept up into multiple SDOs?
    - Should we try to describe volatile adversary infrastructure such as DNS fast flux?
    - 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 embed Cyber Observables directly in the Infrastructure SDO? Or should it reference an Observed Data blob?

2. 2. Potential STIX Cyber Observables Objects

2.1. DNS-related Additions

Additions to the existing Cyber Observable model are in green.



2.2. DNS Query Extension (Network Traffic Object)

Type Name: `dns-request-ext`

The DNS Query request extension specifies a default extension for capturing network traffic properties specific to DNS query requests. The key for this extension when used in the **`extensions`** dictionary **MUST** be `dns-request-ext`.

Properties

Property Name	Type	Description
`qname_ref` (required)	`object-ref`	Specifies a reference to the domain name being queried. The object referenced in this property **MUST** be of type `domain-name`.

qtype (required)	string	Specifies the type of DNS resource record requested.
qclass (optional)	string	Specifies the class of DNS resource record requested.
transaction_id (optional)	string	Specifies the transaction ID value contained in the DNS query message header.
flags (optional)	hex	Specifies the flags contained in the DNS query message header.

2.2.1. Examples

Basic DNS Query (Request)

```
{
  "0": {
    "type": "ipv4-addr",
    "value": "198.51.100.53"
  },
  "1": {
    "type": "domain-name",
    "value": "www.example.com"
  },
  "2": {
    "type": "network-traffic",
    "dst_ref": "0",
    "protocols": [
      "udp",
      "dns"
    ],
    "extensions": {
      "dns-request-ext": {
        "qname_ref": "0",
        "qtype": "A",
        "qclass": "IN"
        }
      }
    }
  }
}
```

Data Exfiltration via DNS Query

```
{
  "0": {
    "type": "ipv4-addr",
    "value": "198.51.100.3"
  },
  "1": {
    "type": "domain-name",
    "value": "FHghlIHgbWFrZSB1cCB0aGUgNjQgY2hhcmFjdGVycyByZXF1aXJlZCBmb3IgYmFzZ.example.com"
  },
  "2": {
    "type": "network-traffic",
    "dst_ref": "0",
    "protocols": [
      "udp",
```

```
      "dns"
    ],
    "extensions": {
      "dns-request-ext": {
        "qname_ref": "0",
        "qtype": "A",
        "qclass": "IN"
        }
      }
    }
  }
}
```

2.3. DNS Query Response Extension (Network Traffic Object)

Type Name: `dns-response-ext`

The DNS Query response extension specifies a default extension for capturing network traffic properties specific to DNS query responses. The key for this extension when used in the **`extensions`** dictionary **MUST** be `dns-response-ext`.

Properties

Property Name	Type	Description
`answer_rr_refs` (optional)	`list` of type `object-ref`	Specifies, via reference, any answer resource records that were returned for a DNS query. The objects referenced in this property **MUST** be of type `dns-record`.
`authority_rr_refs` (optional)	`list` of type `object-ref`	Specifies, via reference, any authority resource records that were returned for a DNS query. The objects referenced in this property **MUST** be of type `dns-record`.
`additional_rr_refs` (optional)	`list` of type `object-ref`	Specifies, via reference, any additional resource records that were returned for a DNS query. The objects referenced in this property **MUST** be of type `dns-record`.
`flags` (optional)	`hex`	Specifies the flags contained in the DNS query response message header.

2.3.1. Examples

Basic DNS Query (Request + Response)

```
{
  "0": {
    "type": "ipv4-addr",
    "value": "192.0.2.1"
  },
  "1": {
    "type": "domain-name",
    "value": "www.example.com"
  },
  "2": {
    "type": "ipv4-addr",
    "value": "203.0.113.2"
  },
  "3": {
    "type": "dns-record",
    "name_ref": "1",
    "record_type": "A",
    "class": "IN",
    "ttl": 300,
    "rdata_refs": ["2"]
  },
  "4": {
    "type": "network-traffic",
    "dst_ref": "0",
    "protocols": [
      "udp",
      "dns"
    ],
    "extensions": {
      "dns-request-ext": {
        "qname_ref": "1",
        "qtype": "A",
        "qclass": "IN"
        },
      "dns-response-ext": {
        "answer_rr_refs": ["3"]
        }
      }
    }
  }
}
```

2.4. DNS Record Object

Type Name: `dns-record`

The DNS Record object represents the properties of a DNS resource record.

2.4.1. Properties

Common Properties

type, description, extensions		
Passive DNS Object Specific Properties		
name_ref, record_type, class, ttl, rdata_refs		
Property Name	**Type**	**Description**
type (required)	string	The value of this property **MUST** be dns-record.
name_ref (required)	object-ref	Specifies the owner name, i.e., the name of the node to which this resource record pertains. The object referenced in this property **MUST** be of type domain-name.
record_type (required)	string	Specifies the type of the resource record.
class (optional)	string	Specifies the class of the resource record.
ttl (optional)	integer	Specifies the number of seconds that the resource record may be cached before the source of the information should again be consulted.
rdata_refs (required)	list of type object-ref	Specifies the data associated with the resource record. Depending on the type of resource record, this property references the appropriate type of Observable Object: • A: The objects referenced in this property **MUST** be of type ipv4-addr. • AAAA: The objects referenced in this property **MUST** be of type ipv6-addr. • CNAME: The objects referenced in this property **MUST** be of type domain-name. • PTR: The objects referenced in this property **MUST** be of type domain-name. • MX: The objects referenced in this property **MUST** be of type domain-name.

2.4.2. Examples

Basic DNS Resource Record

```
{
  "0": {
    "type": "domain-name",
    "value": "example.com"
  },
  "1": {
    "type": "ipv4-addr",
    "value": "203.0.113.2"
  },
```

```
  "2": {
    "type": "dns-record",
    "name_ref": "0",
    "record_type": "A",
    "class": "IN",
    "ttl": 300,
    "rdata_refs": ["1"]
  }
}
```

2.4.3. Passive DNS Extension

Type Name: `pdns-ext`

The Passive DNS extension specifies a default extension for capturing DNS record properties specific to passive DNS. The key for this extension when used in the **`extensions`** dictionary **MUST** be `pdns-ext`.

2.4.3.1. Properties

Property Name	Type	Description
`time_first` (optional)	`timestamp`	Specifies the first time that the record / unique tuple (rrname, rrtype, rdata) has been seen by the passive DNS.
`time_last` (optional)	`timestamp`	Specifies the last time that the unique tuple (rrname, rrtype, rdata) record has been seen by the passive DNS.
`origin` (optional)	`string`	Specifies the resource origin of the Passive DNS response.
`count` (optional)	`integer`	Specifies how many authoritative DNS answers were received at the Passive DNS Server's collectors with exactly the given set of values as answers (i.e. same data in the answer set - compare with the uniqueness property in "Mandatory Fields").
`sensor_id` (optional)	`string`	Specifies the sensor information where the record was seen.
`bailiwick` (optional)	`string`	Specifies the best estimate of the apex of the zone where this data is authoritative.
`zone_time_first` (optional)	`timestamp`	Specifies the first time that the unique tuple (rrname, rrtype, rdata) record has been seen via master file import.
`zone_time_last` (optional)	`timestamp`	Specifies the last time that the unique tuple (rrname, rrtype, rdata) record has been seen via master file import.

origin (optional)	`string`	Specifies the resource origin of the Passive DNS response. This field is represented as a Uniform Resource Identifier (URI).

2.4.3.2. Examples

```
{
   "0":{
      "type":"ipv4-addr",
      "value":"203.0.113.2"
   },
   "1":{
      "type":"dns-record",
      "name_ref":"1",
      "record_type":"A",
      "class":"IN",
      "ttl":300,
      "rdata_refs":[
         "0"
      ],
      "extensions":{
         "pdns-ext":{
            "time_first":"2016-04-06T20:07:09.000Z",
            "time_last":"2016-04-06T20:07:09.000Z",
            "count":42
         }
      }
   }
}
```

2.5. Device Object

2.5.1. Other Data Models to research/reference

- OSquery
- Node.js OS data model
- VERIS Asset Enum

Type Name: `device`

The Device Object represents the properties of a hardware device, such as a PC, mobile phone, IOT device, router, etc..

2.5.1.1. Use Cases

- Characterizing vulnerable devices/those targeted by a particular adversary
- Characterizing devices that are part of some malicious or adversary infrastructure
    - E.g., the constituents of a botnet
- Patterning on vulnerable devices for detection/remediation

- Forensic characterization of compromised/affected devices
- Characterizing virtualized infrastructure

2.5.2. Properties

Common Properties		
type, description, extensions		
Device Object Specific Properties		
model_number, manufacturer, serial_number, installed_os_refs, network_address_refs		
Property Name	**Type**	**Description**
type (required)	`string`	The value of this property **MUST** be `device`.
model_number (optional)	`string`	Specifies the model number of the device.
manufacturer (optional)	`string`	Specifies the name of the manufacturer of the device.
firmware_refs (optional)	`list` of type `object-ref`	Specifies a reference to one or more instances of firmware installed on the device. The objects referenced in this list **MUST** be of type `software`.
installed_os_refs (optional)	`list` of type `object-ref`	Specifies a reference to one or more operating systems installed on the device. The objects referenced in this list **MUST** be of type `software`.
network_address_refs (optional)	`list` of type `object-ref`	Specifies, via reference, the IP addresses used by or assigned to the device. The objects referenced in this list **MUST** be of type `ipv4-addr` or `ipv6-addr`.

2.5.3. Examples

Basic Device (Tablet)

```
{
  "0": {
    "type": "device",
```

```
    "model_number":"SM-T800",
    "manufacturer":"Samsung"
  }
}
```

Basic Device (PC)

```
{
  "0": {
    "type": "device",
    "model_number":"MacbookPro11,2",
    "manufacturer":"Apple"
  }
}
```

Router Device

```
{
  "0": {
    "type": "device",
    "model_number":"AirRouter",
    "manufacturer":"Ubiquiti",
  }
}
```

2.6. Firmware Extension (Software Object)

Type Name: `firmware-ext`

The Firmware extension specifies a default extension for capturing properties specific to any type of firmware that may be running on a device. The key for this extension when used in the **`extensions`** dictionary **MUST** be `firmware-ext`

Properties

Property Name	Type	Description
`firmware_type` (optional)	`open-vocab`	Specifies the type of firmware. This property is an open vocabulary and values **SHOULD** come from the `firmware-type-ov`.
`revision` (optional)	`string`	Specifies the particular revision of the firmware, if separate from the version.
`revision_date` (optional)	`timestamp`	Specifies the date/time stamp of the firmware revision.
`serial_number` (optional)	`string`	Specifies the serial number of the physical firmware chip (if applicable).

2.6.1. Examples

Mobile Phone Bootloader

```
{
   "0":{
      "type":"software",
      "name":"HTCBoot",
      "version":"1.03.001",
      "vendor":"HTC",
      "extensions":{
         "firmware-ext":{
            "firmware_type":"bootloader",
            "revision_date":"2011-01-20T16:59:11Z"
         }
      }
   }
}
```

PC BIOS (generic)

```
{
   "0":{
      "type":"software",
      "name":"Award BIOS",
      "version":"6.00PG",
      "vendor":"Phoenix Technologies",
      "extensions":{
         "firmware-ext":{
            "firmware_type":"bios",
            "revision":"2.053.E2",
            "revision_date":"2005-06-23T12:00:00Z"
         }
      }
   }
}
```

PC BIOS (Dell)

```
{
   "0":{
      "type":"software",
      "version":"AO1",
      "vendor":"Dell",
      "extensions":{
         "firmware-ext":{
            "firmware_type":"bios",
            "revision_date":"2007-08-01T12:00:00Z"
         }
      }
   }
}
```

Mac BIOS

```
{
   "0":{
      "type":"software",
      "version":"MBP112.88Z.0138.B21.1612230939",
      "vendor":"Apple Inc.",
      "extensions":{
         "firmware-ext":{
```

```
            "firmware_type":"bios",
            "revision":"138.21.1~1 (B&I)",
            "revision_date":"2016-12-23T12:00:00Z"
         }
      }
   }
}
```

2.6.2. Firmware Type Vocabulary

Type Name: `firmware-type-ov`

An open vocabulary of firmware types.

Vocabulary Value	Description
`bios`	Specifies basic input/output system (BIOS) firmware.
`uefi`	Specifies the unified extensible firmware interface (UEFI), a replacement for the BIOS.
`bootloader`	Specifies bootloader (i.e., the software that first loads on a device, other than a BIOS) firmware.
`controller`	Specifies controller firmware, such as that which resides on a hard disk.
`embedded-operating-system`	Specifies an embedded operating system that functions as firmware.

2.7. Operating System Extension (Software Object)

Type Name: `os-ext`

The Operating System extensions specifies a default extension for capturing properties of a running instance of an operating system (OS).

2.7.1. Properties

Property Name	Type	Description
`type` (required)	`string`	The value of this property **MUST** be `operating-system`.
`family` (optional)	`open-vocab`	Specifies the overarching family that the OS belongs to. This property is an open vocabulary and values **SHOULD** come from the `os-family-ov`.

patch (optional)	`list` of type `string`	Specifies the patch release version of the operating system.
build (optional)	`string`	Specifies the build-specific or variant string associated with the operating system.

2.7.2. Operating System Family Vocabulary

Type Name: `os-family-ov`

An open vocabulary of operating system families.

Vocabulary Value	Description
`windows`	Specifies the Microsoft Windows family of operating systems.
`linux`	Specifies the Linux family of operating systems.
`macos`	Specifies the Apple MacOS family of operating systems.
`ios`	Specifies the Apple iOS family of operating systems.
`android`	Specifies the Google Android family of operating systems.
`freebsd`	Specifies the FreeBSD family of operating systems.
`sunos`	Specifies the Oracle SunOS family of operating systems.
`aix`	Specifies the IBM AIX family of operating systems.

2.7.3. Examples

Windows 7

```
{
   "0":{
      "type":"software",
      "name":"Windows",
      "cpe":"cpe:2.3:o:microsoft:windows_7:-:sp1:x64:*:*:*:*:*:*",
      "version":"7",
      "vendor":"Microsoft",
      "extensions":{
         "os":{
            "family":"windows"
         }
      }
   }
}
```

Custom OS


```
{
   "0":{
      "type":"software",
      "name":"FooOS",
      "version":"1",
      "vendor":"FooCorp",
      "extensions":{
         "os":{
            "family":"linux"
         }
      }
   }
}
```


OS X


```
{
   "0":{
      "type":"software",
      "name":"Mac OS X",
      "version":"10.12.3",
      "vendor":"Apple",
      "extensions":{
         "os":{
            "family":"macos",
            "patch":"3",
            "build":"16D32"
         }
      }
   }
}
```


This is definitely NOT Ivan's Macbook


```
{
  "0": {
    "type": "device",
    "model_number": "MacbookPro11,2",
    "manufacturer": "Apple",
    "firmware_refs": ["1"],
    "installed_os_refs": ["2"],
    "network_address_refs": ["3", "4"]
  },
  "1": {
    "type": "software",
    "version": "MBP112.88Z.0138.B21.1612230939",
    "vendor": "Apple Inc.",
    "extensions": {
      "firmware-ext": {
        "firmware_type": "bios",
        "revision": "138.21.1~1 (B&I)",
        "revision_date": "2016-12-23T12:00:00Z"
      }
    }
  },
  "2": {
    "type": "software",
    "name": "Mac OS X",
    "version": "10.12.3",
    "vendor": "Apple",
    "extensions": {
```

```
      "os": {
        "family": "macos",
        "patch": "3",
        "build": "16D32"
      }
    }
  },
  "3": {
    "type": "ipv4-addr",
    "value": "198.51.100.3"
  },
  "4": {
    "type": "ipv6-addr",
    "value": "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
  }
}
```

2.8. Network Share Extension (File Object)

Type Name: `network-share-ext`

The Network Share Extension captures properties associated with network shares.

2.8.1. Properties

3. Other Potential Additions

3.1. Actions

Actions can be thought of in terms of system state changes and similar dynamic events, such as the creation of a file, modification of a registry key, and termination of a process. Common sources of Actions include sandboxes/dynamic malware analysis tools and related forms of OS-level instrumentation.

Use Cases

- Sandbox/dynamic malware analysis tool characterization
- Log event characterization
- Action-based patterning
    - Behavioral indicators
    - Analytics

Open Questions

- Completeness - should we try to characterize all (or at least the vast majority) of actions, or a instead a limited subset that only covers more common Actions?
- Semantics - how should we specify the semantics of Actions?
    - If we use the “name” property, can we get away with using only the verb and referenced Objects to clearly define the semantics of an action?
        - E.g., a “create” Action that references a File
        - Some verbs are very specific to particular operations - e.g., “kill” or “terminate” is typically used only in the context of a process or thread

- How do we deal with Actions that require semantics which cannot be clearly expressed by the verb/object tuple alone?
  - E.g., “get windows system directory”, “create file alternate data stream”

Properties

Property Name	Type	Description
type (required)	`string`	The value of this field **MUST** be `action`.
id (required)	`object-id`	TODO: copy-pasta normative language from Part 3, Observed Data
timestamp (optional)	`timestamp`	Captures the local or relative time(s) at which the Action occurred or was observed.
subject_refs (optional)	`list` of type `object-ref`	Specifies a set of one or more Cyber Observable Objects constituting the subject of the Action. For example, a Process that initiated an Action would be the *subject* of the Action.
action (required)	`open-vocab`	Specifies the verb defining the Action. The values for this property **SHOULD** come from the `action-ov` vocabulary.
object_refs (optional)	`list` of type `object-ref`	Specifies a set of one or more Cyber Observable Objects constituting the object(s) of the Action. For example, a Mutex that was created would serve as the “object” of a *create* Action.
action_outcome (optional)	`open-vocab`	Specifies whether the attempted Action was successful or not. The values for this field **SHOULD** come from the `action-outcome-ov` vocabulary.
preceding_action_refs (optional)	`list` of type `object-ref`	Specified a set of one or more actions causatively preceding this action.
ensuing_action_refs (optional)	`list` of type `object-ref`	Specified a set of one or more actions causatively ensuing from this action.

Normative text: at least one of subject_refs or object_refs must be present.

{ action: { type: "stix", id: "uuid", subject_refs: ["uuid", ...], verb: "from stix coa open-vocab", object_refs: ["uuid", ...], preceding_action_refs:["uuid", ...], subsequent_action_refs:["uuid", ...]}

{ action: { type: "obs", id: "cybox_id", subject_refs: ["cybox_id", ...], verb: "from observable actions open-vocab", object_refs: ["cybox_id", ...], preceding_action_refs:["cybox_id", ...], subsequent_action_refs:["cybox_id", ...]}

Vocabularies

Action Vocabulary

Type Name: `action-ov`

A non-exhaustive vocabulary of verbs that define particular Actions.

Vocabulary Value	Description
`create`	Specifies an Action that creates a Cyber Observable Object.
`delete`	Specifies an Action that deletes an existing Cyber Observable Object.
`modify`	Specifies an Action that modifies in some way an existing Cyber Observable Object.
`read`	Specifies an Action that reads from a Cyber Observable Object.
`write`	Specifies an Action that writes to a Cyber Observable Object.
`open`	Specifies an Action that opens a Cyber Observable Object for reading, writing, or modification.
`close`	Specifies an Action that closes a previously opened Cyber Observable Object.
`connect`	Specifies an Action that establishes some form of network connection to a Cyber Observable Object.
`disconnect`	Specifies an Action that disconnects from a Cyber Observable Object.
`upload`	Specifies an Action that uploads a Cyber Observable Object to a network location.
`download`	Specifies an Action that downloads a Cyber Observable Object from a

	network location.

Action Outcome

Type Name: `action-outcome-ov`

A non-exhaustive vocabulary of Action outcomes.

Vocabulary Value	Description
`completed`	The Action successfully executed.
`failed`	The Action failed to successfully execute.
`terminated`	The Action terminated its execution.

Examples

Create File Action

```
{
  "0":{
    "type":"action",
    "action":"create",
    "subject_refs":[
      "1"
    ],
    "object_refs":[
      "2"
    ],
    "timestamp":"2016-01-20T12:31:12.12345Z",
    "action_outcome":"completed"
  },
  "1": {
    "type": "process",
    "pid": 1221,
    "name": "fooproc"
  }
  "2":{
    "type":"file",
    "hashes":{
      "MD5":"4472ea40dc71e5bb701574ea215a81a1"
    },
    "size":25536,
    "name":"foo.dll"
  }
}
```

Cuckoo Sandbox -> Actions

The following are a few snippets of a report generated by Cuckoo Sandbox, along with their corresponding representations using the above model.

Cuckoo Behavioral Trace	Cyber Observable Action

<table>
<tr>
<td>

```
{
  "category":"file",
  "status":1,
  "stacktrace":[
  ],
  "api":"NtCreateFile",
  "return_value":0,
  "arguments":{
    "create_disposition":1,
    "file_handle":"0x0000007c",
    "filepath":"C:\\WINDOWS\\system32\\ntos.exe",
    "desired_access":"0x40100080",
    "file_attributes":0,

"filepath_r":"\\??\\C:\\WINDOWS\\system32\\ntos.exe"
,
    "create_options":96,
    "status_info":1,
    "share_access":1
  }
}
```

</td>
<td>

```
{
  "0":{
    "type":"action",
    "action":"create",
    "object_refs":[
      "1"
    ],
    "action_outcome":"completed"
  },
  "1":{
    "type":"file",
    "name":"ntos.exe",
    "parent_directory_ref":"2"
  },
  "2": {
    "type": "directory",
    "path": "C:\\WINDOWS\\System32"
    }
}
```

</td>
</tr>
<tr>
<td>

```
{
  "category":"synchronisation",
  "status":1,
  "stacktrace":[
  ],
  "api":"NtCreateMutant",
  "return_value":0,
  "arguments":{
    "initial_owner":1,
    "desired_access":"0x001f0001",
    "mutant_name":"__SYSTEM__91C38905__",
    "mutant_handle":"0x00000074"
  },
  "time":1493546959.03401,
  "tid":1132,
  "flags":{

"desired_access":"STANDARD_RIGHTS_ALL|STANDARD_RIGHT
S_REQUIRED|DELETE|READ_CONTROL|WRITE_DAC|WRITE_OWNER
|SYNCHRONIZE"
  }
}
```

</td>
<td>

```
{
  "0":{
    "type":"action",
    "action":"create",
    "object_refs":[
      "1"
    ],
    "action_outcome":"completed"
  },
  "1":{
    "type":"mutex",
    "value":"__SYSTEM__91C38905__"
  }
}
```

</td>
</tr>
<tr>
<td>

```
{
  "category":"registry",
  "status":0,
  "stacktrace":[

  ],
  "last_error":487,
  "nt_status":-1073741772,
  "api":"RegOpenKeyExA",
  "return_value":2,
  "arguments":{
    "access":"0x02000000",
    "base_handle":"0x80000002",
    "key_handle":"0x00000000",
```

</td>
<td>

```
{
  "0":{
    "type":"action",
    "action":"open",
    "object_refs":[
      "1"
    ],
    "action_outcome":"completed"
  },
  "1":{
    "type":"windows-registry-key",

"key":"HKEY_LOCAL_MACHINE\\Software\\Microsoft\\W
indows\\CurrentVersion\\Explorer\\Performance"
  }
```

</td>
</tr>
</table>

<table>
<tr>
<td>

<pre>
"regkey":"HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Performance",

"regkey_r":"Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Performance",
    "options":0
  },
  "time":1493546958.90401,
  "tid":1132,
  "flags":{

  }
}
</pre>

</td>
<td>

<pre>
}
</pre>

</td>
</tr>
</table>