Vertex

Synapse Bootcamp

Module 7

Pre-Storm Background

v0.4 - May 2024

Objectives

- Compare the Synapse UI and Storm
- Review the Synapse data model
- Explain the data model namespace
- Understand how knowing the data model relates to using Storm
- Know some advantages of the Storm query language

The Synapse UI and Storm

- So far we have focused on the Synapse UI
  - Make it easy to start using Synapse
  - Help analysts be productive from day one
- In many cases the UI simply runs Storm for you
  - Simplify common analyst tasks
  - Synapse runs on Storm!
- Using Storm directly gives us more flexibility
  - Do anything - not just what is exposed through menus and icons

Storm **complements** the UI and provides an additional, powerful option.

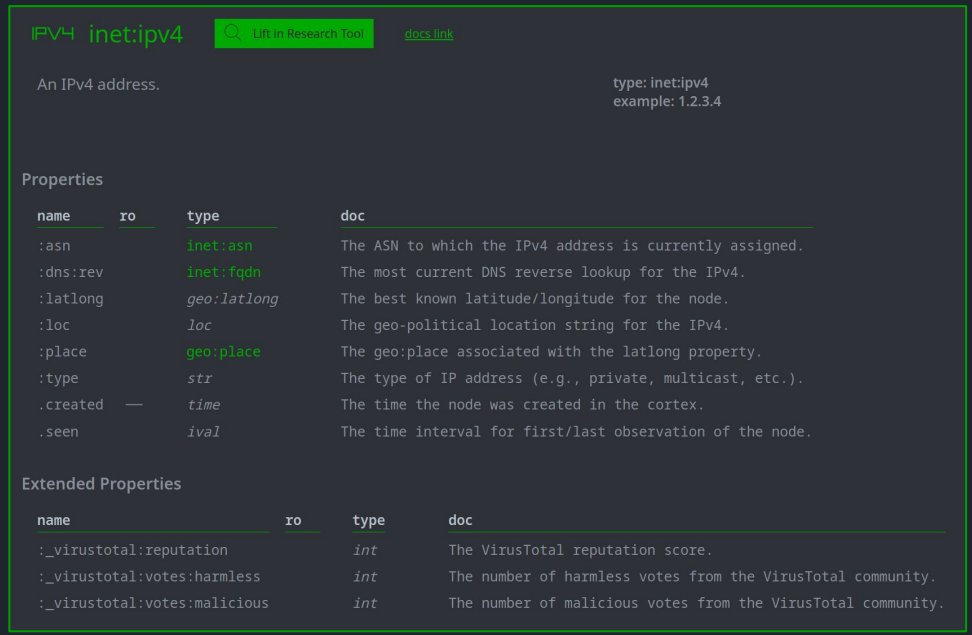
Data Model

Data Model Elements

Forms, Properties, and Nodes

Forms

- "Templates" for creating nodes
  - Consists of **properties** and **types**
- Form name = primary property
  - Secondary properties
  - Universal properties
  - Extended properties



A **form** shows us *how* we can create something. A **node** is the *object we create* based on the form.

Forms as Objects

- Forms can represent **"things"** in the real world (including abstract things)

Object	Synapse Form
An IP address	`inet:ipv4` or `inet:ipv6`
An article, paper, report, or document	`media:news`
A server certificate or code-signing certificate	`crypto:x509:cert`
A computer or device	`it:host`
An organized group of individuals	`ou:org`
A commercial industry	`ou:industry`
A set of contact information	`ps:contact`

Forms as Relationships

- Forms can represent **relationships** between objects:

Object	Synapse Form
A DNS A record (FQDN and the IP it resolves to)	`inet:dns:a`
A certificate associated with a host (IP / port)	`inet:tls:servercert` / `inet:tls:clientcert` (new) `inet:ssl:cert` (old)
A file signed with a code signing certificate	`crypto:x509:signedfile`
A URL hosting a file	`inet:urlfile`
A file existing in a particular location	`file:filepath`
An email message with an attached file	`inet:email:message:attachment`
A person attending a conference	`ou:attendee`

Forms as Events

- Forms can represent **events** or **instances** of things
  - Typically include a `:time` property

Object	Synapse Form
A specific DNS query and / or response	`inet:dns:request`, `inet:dns:answer`
A specific HTTP query	`inet:http:request`
A file / process interacting with a file system	`it:exec:file:read`, `it:exec:file:add`, `it:exec:file:write`, `it:exec:file:del`
A conference	`ou:conference`
An alert	`risk:alert`
An attack	`risk:attack`

Forms and Primary Properties

- A form's primary property must be **unique** for that form
  - Helps ensure Synapse does not store multiple copies of a piece of data

Primary Property Category	Description	Example
Simple	Single value	`inet:fqdn = vertex.link` `inet:ipv4 = 1.2.3.4`
Composite (comp)	Two values that make a unique pair	`inet:dns:a = (vertex.link,1.2.3.4)`
Guid (globally unique identifier)	128-bit value	`media:news = 6b37b042c41ea0b14aa62a42e7a4a699`

Why Guids?

- When it's impossible to "uniquely" describe an object with a set of values
  - What makes a "DNS request" unique?
- When the amount of data you have about something may vary

inet:dns:request

Lift in Research Tool

docs link

A single instance of a DNS resolver request and optional reply info.

type: inet:dns:request
base: guid

Properties

name	ro	type	doc
:exe		file:bytes	The file containing the code that attempted the DNS lookup.
:host		it:host	The host that attempted the DNS lookup.
:proc		it:exec:proc	The process that attempted the DNS lookup.
:query		inet:dns:query	A DNS query unique to a given client.
:query:name		*inet:dns:name*	A DNS query name string. Likely an FQDN but not always.
:query:name:fqdn		inet:fqdn	A Fully Qualified Domain Name (FQDN).
:query:name:ipv4		inet:ipv4	An IPv4 address.
:query:name:ipv6		inet:ipv6	An IPv6 address.
:query:type		int	The base 64 bit signed integer type.
:reply:code		int	The DNS server response code.
:server		inet:server	A network server address.
:time		time	A date/time value.
.created	—	time	The time the node was created in the cortex.
.seen		ival	The time interval for first/last observation of the node.

Forms vs. Nodes

Form (`inet:ipv4`)

Form / Property	Type
inet:ipv4	inet:ipv4
inet:ipv4.seen	ival
inet:ipv4.created	time
inet:ipv4:asn	inet:asn
inet:ipv4:dns:rev	inet:fqdn
inet:ipv4:latlong	geo:latlong
inet:ipv4:loc	loc
inet:ipv4:place	geo:place
inet:ipv4:type	str

Node (`inet:ipv4=60.248.52.95`)

Form / Property	Value
inet:ipv4	60.248.52.95
inet:ipv4.seen	(2018/05/18 09:48:15.000, 2018/05/18 09:48:15.001)
inet:ipv4.created	2021/08/18 20:51:47.651
inet:ipv4:asn	3462
inet:ipv4:dns:rev	60-248-52-95.hinet-ip.hinet.net
inet:ipv4:latlong	23.7113,120.3897
inet:ipv4:loc	tw.yun.yunlin
inet:ipv4:place	04f520ec542de2ee87d10bb083ac081c
inet:ipv4:type	unicast

The Data Model and Storm

Why All This Data Model Stuff?

- Storm allows you to "ask questions" of your Synapse data
- The data model is the "language" you use to tell Storm what to do

Question	Storm Example
"What do we know about the FQDN evil.com?"	`inet:fqdn=evil.com`
"What IP addresses has evil.com resolved to?"	`inet:fqdn=evil.com -> inet:dns:a -> inet:ipv4`
"What files communicate with evil.com?"	`inet:fqdn=evil.com -> inet:dns:request -> file:bytes`

Storm "Vocabulary"

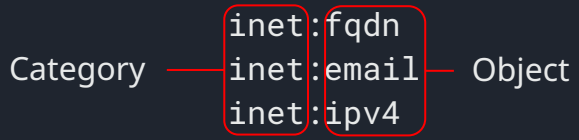
Object	Example
Form	`inet:ipv4`
Form and value	`inet:ipv4 = 217.174.156.100`
Form and property	`inet:ipv4:asn`
Form and property and value	`inet:ipv4:asn = 9009`
Tag (as a node)	`syn:tag = rep.eset.sednit`
Tag (as a label on a node)	`#rep.eset.sednit`

Let's look more closely at how we name forms/properties (and why)!

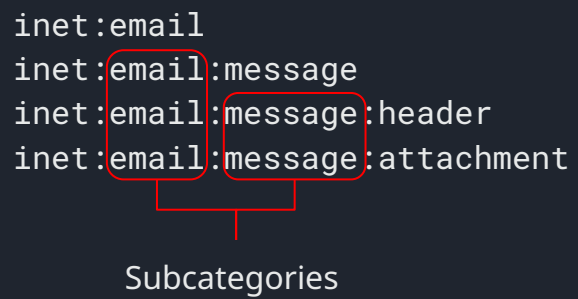
Form Namespace

- At least two elements separated by a colon (:)
  - Category: broadly groups related forms
  - Name: the form (object) within that namespace

Simple namespace:



May include subcategories:



Form Wildcards

- Use the wildcard (*) to match one or more namespace elements
  - Instead of:

```
hash:md5 hash:sha1 hash:sha256 hash:sha512
```

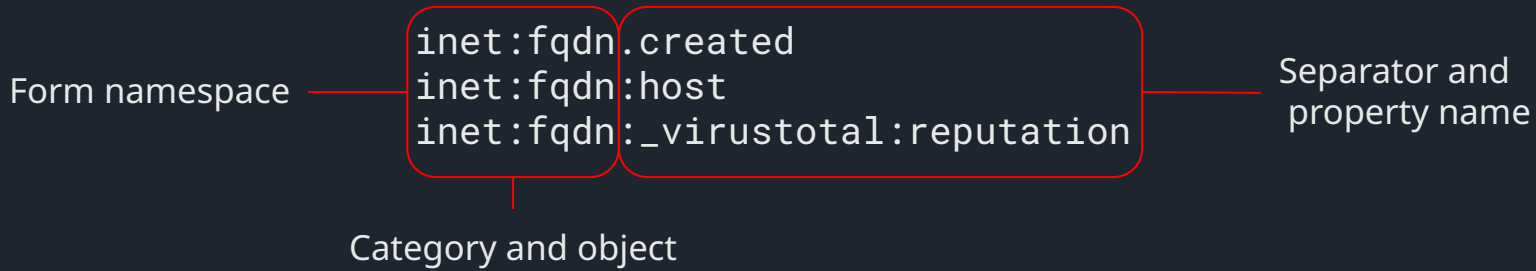
  - Can use:

```
hash:*
```

Tip: The wildcard is a specialized use case, but handy when you need it!

Property Namespace

- Properties exist within the form's namespace
  - Different **separator** depending on the kind of property



- Properties may have their own hierarchies:

```
file:bytes
file:bytes:md5
file:bytes:mime
file:bytes:mime:pe:compiled
file:bytes:mime:pe:imphash
```

Properties: Full vs Relative

- **Full property name:** form and property names together
  - `file:bytes.created`
  - `inet:fqdn:host`
  - `inet:dns:request:query:name:ipv4`
  - `inet:ivp4:_virustotal:reputation`

- **Relative property name:** the property name alone
  - `.created`
  - `:host`
  - `:query:name:ipv4`
  - `:_virustotal:reputation`

Understanding form and property naming conventions will make it much easier to use Storm!

Types

Types

- Every property has a **type**
  - What kind of data does Synapse expect?
  - `type = inet:fqdn`
    - A value that looks reasonably like a domain / FQDN
- Types provide three key advantages:
  - **Type enforcement:** how Synapse helps keep data **tidy**
  - **Type-specific behavior:** how Synapse helps you **work with** data
  - **Type awareness:** how Synapse helps you **navigate** the data

Many data storage and analysis systems only use a very simple set of types - for example, integers, Booleans, and strings.

Form / Property	Type
inet:ipv4	`inet:ipv4`
inet:ipv4.seen	`ival`
inet:ipv4.created	`time`
inet:ipv4:asn	`inet:asn`
inet:ipv4:dns:rev	`inet:fqdn`
inet:ipv4:latlong	`geo:latlong`
inet:ipv4:loc	`loc`
inet:ipv4:place	`geo:place`
inet:ipv4:type	`str`

Type Enforcement

- Defines how data should "look" in Synapse
- In many systems, a URL is simply a **string**
  - Potentially allows creation / input of invalid URLs

```
fd8q3:523/pogostick&?derp=hahahaha.net
```

```
visi@vertex.link
```

- In Synapse, a URL is an **inet:url**
  - Must include a protocol header - string followed by **://**
  - Must include a valid hostname (FQDN or IP address)
  - Prevent a lot of bad data (though not all)

Protip: If you see a `BadTypeValu` error, something tried to set an invalid property *value*, given the property's *type*.

Type-Specific Behavior

- Custom optimizations for some data types
- An IPv4 address (`inet:ipv4`) is a 32-bit binary number
  - 01111011011110000110111010010000
- Synapse **stores** IPv4 addresses as decimal integers
  - 2071490192
- Synapse **shows us** (represents) an IPv4 as dotted-decimal
  - 123.120.110.144
- Synapse allows us to **enter** an IPv4 address as:
  - Dotted decimal, decimal integer, hexadecimal, IP range, CIDR block...

Protip: Many optimizations "just work". Some make things easier when using Storm. See the Storm documentation for details.

Type Awareness

- Synapse is "aware" of the **type** of each property
  - Allows Synapse to "understand" and **navigate** the data model
- Most objects (nodes) are not explicitly linked
- If I ask about a specific object:
  - `inet:ipv4 = 27.102.106.149`
- Synapse can show me all the other nodes where that IP is a **property**:
  - DNS A records (`inet:dns:a`)
  - Network flows (`inet:flow`)
  - URLs (`inet:url`)
  - SSL / TLS certificates (`crypto:x509:cert`)

Protip: **Type awareness** is a big part of how "Explore" in Synapse (or pivoting in Storm) works!

Type Awareness

- Nodes with **properties** with the same **type** and **value** are **connected**

Form / Property	Value
inet:fqdn	blog.cdsend.xyz
:domain	cdsend.xyz
:host	blog
:issuffix	False
:iszone	False
:zone	cdsend.xyz

Form / Property	Value
inet:dns:a	(blog.cdsend.xyz, 91.195.240.12)
:fqdn	blog.cdsend.xyz
:ipv4	91.195.240.12

Form / Property	Value
inet:ipv4	91.195.240.12
:asn	47846
:loc	de
:type	unicast

Protip: Type awareness is how "Explore" in Synapse (or pivoting in Storm) works!

Light Edges

Light Edges

- Used to **connect** nodes without a property-based relationship
  - A public report (`media:news`) that contains IOCs (such as `hash:md5`)
  - A source (`meta:source`) that provided data (such as `file:bytes`)
- Light edges have:
  - A **name** ("verb")
    - Describes the edge (e.g., "refs" for "references")
    - Distinguish different types of relationships
  - A **direction** (like an arrow)
    - The relationship only makes sense one way
    - ✅ "The article references the MD5 hash"
    - ❌ "The MD5 hash references the article"

```
media:news:publisher:name=eset -(refs)> inet:fqdn=evil.com
```

The Synapse UI and Storm

- What does Storm unlock for us?

Synapse UI	Storm
Query bar: Lookup / Text Search modes	Query bar: Storm mode
Easily work with **common indicators**	Work with **any element** of the data model
Easy, broad-based navigation - **discover / explore**	Narrowly target your research - **precisely answer specific questions**
Quick access to **common** Power-Ups / commands	**Customize** execution when needed
No automation	Leverage for **automation!**

Protip: Most analysts use a **combination** of UI elements and Storm, based on need and preference!

Summary

- The key components of Synapse's data model are **forms, properties, nodes, types,** and **light edges**
- Types support **type enforcement, type-specific behavior,** and **type awareness**
- Forms and properties use a **structured namespace** with most elements separated by a colon (:)
- The data model is a large part of the vocabulary used with **Storm**
- Using **Storm** complements the Synapse UI and gives you additional power and flexibility