



Vertex

Synapse Bootcamp

Module 10

Filtering in Storm

v0.4 - May 2024



Objectives

- Understand how **filter** operations work in Storm
- Know how to **filter** nodes using Storm
 - Simple filters
 - Filters using mathematical comparisons
 - Filters using Synapse's extended filter operators
- Know what a **compound filter** is in Storm
- Know what a **subquery filter** is in Storm



Filtering Data with Storm



Storm Operations

| Operation | Meaning | Common Storm Operator | UI Equivalent |
|---------------|----------------------------------------------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------|
| Lift | Select data (nodes) from Synapse | Query bar - Storm | Query bar - Lookup / Text Search query and copy menu options |
| Pivot | Move between nodes that share the same property value | -> or <- * | Explore button pivot menu option |
| Traverse | Move between nodes that are linked by an edge | -(*)> or <(*)- | Explore button |
| Filter | Include / exclude a subset of nodes | + or - | n/a (column filters; query / select menu options) |
| Run | Execute a Storm command | <command> | Node Action |
| Modify / Edit | Modify or delete properties Add or remove tags Add nodes | [] or [()] | Inline property edit; delete menu option Add / remove tags menu options Lookup or Auto Add / Add Node |



Filtering in Storm

- Keep or discard a **subset** of your current data
- Filtering in Storm requires:
 - + or - symbol
 - Include or exclude
 - The data you want to filter
- Tell Synapse what to filter using:
 - Form / property / value
 - Tag
 - Form wildcard
 - Interface

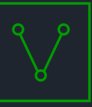
Tip: most of the methods used to lift nodes are also used to filter them. If you can do one, you can do both!



Basic Filters

| Kind of Filter | Example | Operation |
|-----------------------------|-----------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| By form | <code>+inet:ipv4</code> | Include only IPv4s |
| By secondary property | <code>+inet:ipv4:asn</code> | Include only IPv4s that have an <code>:asn</code> property |
| By tag | <code>-#rep.mandiant.ap1</code> | Exclude nodes with this tag |
| By form wildcard | <code>+hash:*</code> | Include only <code>hash:</code> forms (<code>hash:md5</code> , <code>hash:sha1...</code>) |
| By primary property value | <code>-inet:ipv4 = 1.2.3.4</code> | Exclude the IPv4 <code>1.2.3.4</code> |
| By secondary property value | <code>+inet:ipv4:asn = 9009</code> <code>+:asn = 9009</code> | Include only IPv4s with AS 9009 |

If you notice a similarity between lift and filter operations, that's because they are nearly identical!



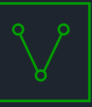
Filters and Relative Properties

- In many cases, Storm supports the use of **relative property names**
- Full property name (form and property):

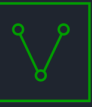
```
file:bytes#rep.carbonblack.ap28 +file:bytes:mime:pe:pdbpath
```

- Relative property name (property name only):

```
file:bytes#rep.carbonblack.ap28 +:mime:pe:pdbpath
```



Filter Demo - Basic Filters



Additional Filter Operators



Additional Filter Operators

| Symbol(s) | Type of Operator | Example |
|-----------------------------|-------------------------|-------------------------------------------|
| +/- with =, >, <, >=, <= | Standard / Mathematical | -file:bytes:size < 1024 |
| +/- with ~= | By regular expression | +ou:name ~= bank |
| +/- with ^= | By prefix | +inet:ipv4:loc ^= cz |
| +/- with @= | By time / interval | +inet:dns:request:time @= (now, '-1 day') |
| +/- with * [<operator>] | Arrays | -it:mitre:attack:group:names* [^=temp] |
| +/- with (and or not) | Compound filter | +((hash:md5 and #cno) or inet:fqdn) |
| +/- with { <query> } | Subquery filter | +{ -> crypto:x509:signedfile } |



Filter Demo - Additional Filters



Specialized Filters



Compound Filters

- Allow you to **combine** multiple filter criteria using logical operations
 - o **AND, OR, NOT**

```
inet:ipv4#rep +( :asn=24940 or :loc^=us )
```

```
inet:ipv4#rep +( :asn=24940 and not :loc^=us )
```

- Use parentheses to clarify order or grouping

```
inet:ipv4#rep +( :asn=20473 or ( :loc^=cn or :loc^=tw ) )
```

```
inet:ipv4#rep +( ( :asn=13768 and :loc^=ca ) or :loc^=us )
```



Subquery Filters

- Conceptually all filters are just filters:

```
inet:fqdn -> inet:dns:a +<filter goes here>
```

- A regular filter evaluates a form, property, or tag on the inbound nodes:

```
inet:fqdn -> inet:dns:a +.seen
```

- A compound filter evaluates multiple forms / properties / tags:

```
inet:fqdn -> inet:dns:a +(:ipv4=4.4.4.4 or :ipv4=8.8.8.8)
```

- In a subquery filter, the filter is a **Storm query**:

```
inet:fqdn -> inet:dns:a +{ <storm_goes_here> }
```



Subquery Filters

- **Example:**
 - You want to view a set of **DNS A** records for a domain (`inet:dns:a nodes`)
 - You **don't** want to see records where the **IP address** is:
 - Private (`-inet:ipv4:type=private`)
 - A sinkhole (`-#cno.infra.dns.sink.hole`)
- The things you want to filter on are properties of the IP addresses
 - ...not the DNS A records themselves

How can I use Storm to just view the DNS A records I care about?



Subquery Filters

- A subquery filter can be thought of as a "what if" operation
 - o Performs Storm operations "in the background" without affecting your current results
- For our DNS A example...
 - o "What if" I looked at the associated IPv4 addresses?
 - -> `inet:ipv4`
 - o ...are any of them non-routable?
 - `-:type=private`
 - o ...are any of them sinkholes?
 - `-#cno.infra.dns.sink.hole`

We want to filter our DNS A records based on what the **adjacent** `inet:ipv4` nodes look like.



Use a Subquery Filter

- We want to filter out certain IPv4s, but view the DNS A records:

```
inet:fqdn#rep.mandiant.apt1 -> inet:dns:a -> inet:ipv4 -:type=private  
-#cno.infra.dns.sink.hole
```

- With a subquery filter we can do that!

```
inet:fqdn#rep.mandiant.apt1 -> inet:dns:a +{ -> inet:ipv4 -:type=private  
-#cno.infra.dns.sink.hole }
```



Subquery Filter Demo



Subquery Filters

- Subquery filters can also be used with mathematical operators
 - o Compare the **filter result** using =, <, >, <=, >=
- Find:
 - o Files (file:bytes)...
 - o ...that are malicious (#rep)...
 - o ...and were created in the past 24 hours (.created@=(now, '-1 day'))
 - o ...that are detected as malicious **by 10 or fewer antivirus engines** (subquery filter!)

```
file:bytes#rep +.created@=(now, '-1 day')
+{ -> it:av:scan:result +:verdict=malicious }<10
```



Synapse UI and Storm

| Synapse UI | Storm Filters |
|---------------------------------------------------------------------------------|-------------------------------------------------|
| Pseudo-filter with column filters (hides results) | True filtering |
| Pseudo-filter query and select menu options (resets / runs a new query) | True filtering |
| Limited ability to construct compound filters (with column filters) | Full ability to use Boolean logic for filtering |
| No ability to create subquery filters | Powerful subquery filter capabilities |



Summary

- **Filter** operations in Storm are nearly identical to **lifts**
- A few custom filters give you more power and flexibility
- **Compound** filters combine:
 - o Multiple filter criteria
 - o Using logical operators (not, and, or)
- **Subquery filters** allow you to:
 - o Filter your **current** results based on properties or tags of **nearby** nodes
 - Use Storm to "look ahead" to other nodes
 - Decide to keep or discard **current** nodes based on what you find