**Directory Structure**

… 5G-hybrid-threats/

├── ontology/

│ ├── ontology/

│ │ ├── ontology.ttl # Main ontology file

│ │ ├── shapes.ttl # Original SHACL shapes file

│ │ ├── shapes\_after\_update.ttl # Updated SHACL shapes file with impactLevel scenario

│ │ ├── validate\_shacl.py # Validation script for original shapes.ttl

│ │ ├── validate\_shacl\_after\_update.py # Validation script for shapes\_after\_update.ttl

├── Documents/ # New directory for documentation

│ ├── impact\_scenario.docx # Documentation for the impactLevel scenario

\*\*\*

**Understanding SHACL Shapes in the 5G Hybrid Threat Ontology**

The SHACL (Shapes Constraint Language) file, ‘shapes.ttl’, is critical in defining constraints and validation rules for the ontology. It is divided into four main categories of SHAPES:

**Categories of SHAPES:**

1. **Node Shapes**  
   Define constraints at the class level, specifying properties and their constraints for instances of a given class.
2. **Property Shapes**  
   Define specific constraints for individual properties, such as datatype restrictions or cardinality.
3. **Reusable Constraints**  
   Modular and reusable rules that can be applied across multiple Node Shapes or Property Shapes.
4. **Targeted Validation Shapes**  
   Highly specific SHAPES designed for validating particular scenarios, often using targeted classes or properties.

**Node Shapes (Detailed Example)**

Node Shapes specify constraints for instances of a given class. Below is an example of a Node Shape for the Asset class:

@prefix ex: <http://example.org/5g-security#> .

@prefix sh: <http://www.w3.org/ns/shacl#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:AssetShape a sh:NodeShape ;

sh:targetClass ex:Asset ;

sh:property [

sh:path ex:owner ;

sh:datatype xsd:string ;

sh:minCount 1 ;

sh:message "Every asset must have an owner." ;

] ;

sh:property [

sh:path ex:impactLevel ;

sh:datatype xsd:string ;

sh:in ("High" "Medium" "Low") ;

sh:message "Asset impact level must be specified as High, Medium, or Low." ;

] .

**Explanation of Key Elements:**

* sh:targetClass ex:Asset: Specifies that the shape applies to instances of the Asset class.
* sh:path ex:owner: Defines a required property, owner, for Asset instances.
* sh:datatype xsd:string: Specifies the owner property must be a string.
* sh:minCount 1: Enforces that at least one owner must be specified.
* **sh:path ex:impactLevel: Defines another property, impactLevel, with additional constraints**.
* sh:in ("High" "Medium" "Low"): Ensures impactLevel must be one of the specified options.

**Integration with the ‘shapes.ttl’ File**

Consider the following categories from the AssetShape-NodeShape code above:

ex:AssetShape a sh:NodeShape ;

sh:targetClass ex:Asset ;

sh:property [

sh:path ex:owner ;

sh:datatype xsd:string ;

sh:minCount 1 ;

sh:message "Every asset must have an owner." ;

] ;

These link directly to corresponding property definitions in the ‘shapes.ttl’ file. The specific links include ‘AssetShape’ and ‘NodeShape’ and ‘targetClass’ that appear in the ‘shapes.ttl’ file and are identified below within **[[bold double square brackets]].** Consider the opening of the 1356 line ‘shapes.ttl’ file:

**[[ <http://example.org/5g-security#AssetShape> a sh:NodeShape ; ]]**

**[[ sh:property <http://example.org/5g-security#Asset\_affectsAssetProperty>, ]]**

<http://example.org/5g-security#Asset\_enablesProperty>,

<http://example.org/5g-security#Asset\_exploitsProperty>,

<http://example.org/5g-security#Asset\_hasAcceptableLevelProperty>,

<http://example.org/5g-security#Asset\_hasIndicatorProperty>,

<http://example.org/5g-security#Asset\_hasRiskLevelProperty>,

<http://example.org/5g-security#Asset\_hasVulnerabilityProperty>,

**<http://example.org/5g-security#Asset\_impactLevelProperty>, *{*** *Line 12* ***}***

<http://example.org/5g-security#Asset\_impactProperty>,

<http://example.org/5g-security#Asset\_impactsProperty>,

<http://example.org/5g-security#Asset\_isMitigatedByProperty>,

<http://example.org/5g-security#Asset\_isOwnedByProperty>,

<http://example.org/5g-security#Asset\_isReducedByProperty>,

<http://example.org/5g-security#Asset\_lastAssessmentProperty>,

<http://example.org/5g-security#Asset\_likelihoodProperty>,

<http://example.org/5g-security#Asset\_mitigatesProperty>,

<http://example.org/5g-security#Asset\_protectsProperty>,

<http://example.org/5g-security#Asset\_requiresControlProperty>,

<http://example.org/5g-security#Asset\_requiresMitigationProperty>,

<http://example.org/5g-security#Asset\_riskLevelProperty>,

<http://example.org/5g-security#Asset\_severityProperty>,

<http://example.org/5g-security#Asset\_threatensProperty> ;

**[[ sh:targetClass <http://example.org/5g-security#Asset> . ]]**

The AssetShape-NodeShape code above also links with line 254 of the ‘shapes.ttl’ file with the category “Asset\_affectsAssetProperty.”

<http://example.org/5g-security#Asset\_affectsAssetProperty> a sh:PropertyShape ;

    sh:message "Instances of http://example.org/5g-security#Asset must have at least one value for http://example.org/5g-security#affectsAsset." ;

    sh:minCount 1 ;

    sh:path <http://example.org/5g-security#affectsAsset> .

For a complete understanding of the SHACL file, the three categories of 2. **Property Shapes**, 3. **Reusable Constraints**, and 4. **Targeted Validation Shapes** can be examined similarly. Each category serves a unique purpose in ensuring the ontology's compliance and validation effectiveness.

**The role of the ‘impactLevel’ in the ‘shapes.ttl’ file**

In the 5G Hybrid Threat SHACL file, i.e., ‘shapes.ttl’, **NodeShapes** define constraints for specific classes. They describe which properties instances of a given class must or can have, their acceptable values and other constraints. Below, we explore how these constraints are represented and highlight a specific example related to impactLevel.

**Example: Asset Shape - Node Shape**

We have seen from the AssetShape-NodeShape code above how NodeShape for the Asset class is defined. We have also witnessed the positioning of ‘impactLevel’ in the opening lines of the ‘shapes.ttl’ file.

**[[ <http://example.org/5g-security#AssetShape> a sh:NodeShape ; ]]**

**…**

**<http://example.org/5g-security#Asset\_impactLevelProperty>, *{*** *Line 12* ***}***

**Drilling Down: Impact Level Property**

The impactLevel property of an Asset is referenced further on in the ‘shapes.ttl’ file at **lines 289–292**:

<http://example.org/5g-security#Asset\_impactLevelProperty> a sh:PropertyShape ;

sh:message "Instances of http://example.org/5g-security#Asset must have at least one value for http://example.org/5g-security#impactLevel." ;

sh:minCount 1 ;

sh:path <http://example.org/5g-security#impactLevel> .

This PropertyShape enforces a minimum cardinality of 1 for the impactLevel property and provides a validation message.

To enhance clarity, we can explicitly define the acceptable values for impactLevel (High, Medium, and Low) by extending the SHACL file. For example, the impactLevel property of an Asset is now detailed in the ‘shapes\_after\_update.ttl’ file at **lines 259–265**:

<http://example.org/5g-security#Asset\_impactLevelProperty> a sh:PropertyShape ;

sh:message "Instances of http://example.org/5g-security#Asset must have at least one value for http://example.org/5g-security#impactLevel." ;

sh:minCount 1 ;

sh:path <http://example.org/5g-security#impactLevel> ;

sh:in ("High" "Medium" "Low") ;

sh:datatype xsd:string ;

sh:message "Impact level must be one of High, Medium, or Low." .

**Explanation of High, Medium, and Low Levels**

**Context**

The impactLevel defines the severity of an asset’s impact if compromised. By categorizing this property into discrete levels, we ensure:

* **Uniformity:** Standardized severity levels facilitate comparison across assets.
* **Validation:** Prevents the use of invalid values for impactLevel.
* **Clarity:** Makes the ontology user-friendly for analysis and reporting.

**Enhancing the SHACL File**

The updated SHACL file now provides a more complete representation of the impactLevel property for Asset instances. By introducing the sh:in constraint, users and validation tools will automatically enforce the use of valid values (High, Medium, or Low).

**Impact on the Ontology**

This modification improves the ontology’s utility for:

1. **Validation:** Ensures consistency in asset impact definitions.
2. **Interoperability:** Aligns with common categorization practices.
3. **Clarity:** Helps users understand constraints at a glance.

**\*\*\***

**Further Considerations**

1. Example of Validation and Invalid Scenarios

After integrating sh:in, validation ensures only predefined values are allowed:

* **Valid:** impactLevel: "High"
* **Invalid:** impactLevel: "Critical"

2. Broader Applications

* **Risk:** A Threat could have riskLevel values of High, Medium, or Low.
* **Likelihood:** Likelihood values could follow a similar breakdown.

3. Running SHACL Validation

Users can run the SHACL validation using:

* validate\_shacl\_after\_update.py
* The modularized shapes\_after\_update.ttl file.

This ensures proper validation and enforces constraints defined within the SHACL file. Details as to how to run the validation script are available in the README.md file contained in the …5G-hybrid-threats/ontology directory.