ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Том 110

ANNUAL OF SOFIA UNIVERSITY “ST. KLIMENT OHRIDSKI”
FACULTY OF MATHEMATICS AND INFORMATICS
Volume 110

CAPEC ONTOLOGY

VLADIMIR DIMITROV

CAPEC is an effort coordinated by MITRE Corporation. Its aim is attack pattern database structured in taxonomies. CAPEC is available as XML document from its project site. CAPEC structure and content are under permanent change and development. It is still not mature database but may be never will.
CAPEC, CWE, and CVE are databases devoted to attacks, weaknesses, and vulnerabilities. They refer each other forming a knowledge ecosystem in cybersecurity area.
Traditional approach for knowledge presentation as information does not work well with conceptualizations under dynamics of this ecosystem and particularly of CAPEC. In this paper an alternative approach to CAPEC knowledge presentation is proposed, as ontology. First, CAPEC structure and content are discussed and then ontology structure is introduced. CAPEC as ontology opens doors to “open world” concept that is more adequate to ecosystem dynamics.

Keywords: cybersecurity, attack patterns, ontology, CAPEC, OWL

CCS Concepts:
• Security and privacy∼Formal methods and theory of security∼Formal security models;
• Security and privacy∼Formal methods and theory of security∼Logic and verification

1. Introduction

CAPEC (Common Pattern Enumeration and Classification) [2] is an effort coordinated by MITRE Corporation. Its aim is attack pattern database structured in taxonomies.

CAPEC is freely available in XML format from the site.

Some basic terms from [2]:

- “An attack pattern is a description of the common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. Attack patterns define the challenges that an adversary may face and

DOI: 10.60063/GSU.FMI.110.63-83

how they go about solving it. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples."

- "An attack pattern is the common approach and attributes related to the exploitation of a weakness in a software, firmware, hardware, or service component."
- "An attack (noun) is the use of an exploit(s) by an adversary to take advantage of a weakness(s) with the intent of achieving a negative technical impact(s). An attack is part of the bigger "Cyber Attack Lifecycle" that includes the following tasks: reconnaissance, weaponize, deliver, exploit, control, execute, and maintain."
- "An exploit (noun) is an input or action designed to take advantage of a weakness (or multiple weaknesses) and achieve a negative technical impact. The existence (even if only theoretical) of an exploit is what makes a weakness a vulnerability."

Attack pattern is an abstraction mechanism that describes execution of known attack. It is a blueprint for an exploit; has applicability context and recommended attack mitigation methods.

Attack pattern concept comes from design patterns [1] in object-oriented design, but in destructive context.

Design patterns are common design problem solutions at high level. They cannot be used directly for codding. Instead, design pattern is a description how the problem can be solved.

However, attack patterns are not very abstract. They are related to some functionality type and some exploited weaknesses.

Attack patterns are not very specific related to some concrete application.

Note 1. The term "structured text" below is plain text – literal structured in accordance with some rules. It does not contain special markers.

Note 2. CAPEC elements are defined with XSD types. It is more readable to present elements by names but not by types. In [2] schema is presented by types but only element names are visible in CAPEC catalog – not the types. That is the presentation way used in next section.

2. CAPEC structure

CAPEC catalog (Attack_Pattern_Catalog) contains elements Views, Categories, Attack_Patterns, and External_References. The last ones contain elements View, Category, Attack_Pattern, and External_Reference correspondingly.

Attack_Pattern_Catalog have attributes Name, Version, and Date – required.

2.1. CAPEC structure

View organizes attack patterns from specific point of view. It has attributes ID, Name, Type, and Status – required.

There are three type of views: Explicit, Implicit, and Graph. The catalog currently contains Implicit and Graph views.

Explicit view simply enumerates its members.

Implicit view dynamically defines its members via XPath query.

Graph view has hierarchical structure. Graph views have as children only categories in the current catalog, and the categories have only meta attacks. It is not controlled by CAPEC XSD schema.

CAPEC-533 is obviously an explicit view but is represented as implicit one – its members are simply enumerated.

The view contains elements discussed below.

Objective element is a required structured text describing the view perspective.

Audience element is a list of Stakeholder elements. Every Stakeholder element has two required element: Type and Description. Type element is enumeration with values: Academic Researchers, Applied Researchers, Assessment Customers, Assessment Vendors, CAPEC Team, Educators, Information Providers, Software Customers, Software Designers, Software Developers, Software Vendors, and Other. Description element is a structured text.

Filter element set XPath query for implicit views.

References element is sequence of Reference elements. These are external references.

Reference element has External_Reference_ID and Section attributes. First attribute is in REF-n format – something like footnote. Section points to specific location in the reference.

Notes element is list of Note elements. This is additional information of any kind.

Note element is structured text. Its Type attribute may have as value one of Maintenance, Relationship, Research Gap, Terminology or Other.

Content_History element contains catalog history. Its elements are Submission, Modification, Contribution, and Previous_Entry_Name.

Submission element has elements Submission_Name, Submission_Organization, Submission_Date, and Submission_Comment. Except Submission_Date that is date, all they are strings.

Modification element has elements Modification_Name, Modification_Organization, Modification_Date, Modification_Importance, and Modification_Comment. All they are strings except Modification_Importance that can has as value Normal or Critical.

Contribution element has elements Contribution_Name, Contribution_Organization, Contribution_Date, and Contribution_Comment. Its required attribute Type can have as value Content or Feedback.

Previous_Entry_Name element is a string with required attribute Date.

Members element links explicit and graph views with its members. In graph views, Members points only to categories. There are no explicit views in CAPEC catalog.

There is Relationships element in Categories with same type and purpose as Members in Views.

Members element may contain arbitrary number of Member_Of and Has_Member elements. There are no restrictions on view members in CAPEC XSD schema, but in the catalog contents, graph views have categories for members and Relationships in categories – only attack patterns. Implicit views have as members attack patterns.

Members_Of and Has_Member elements have the same structure. CAPEC_ID attribute can point to any catalog entry by CAPEC XSD schema, but in practice, it is not true – it is discussed below.

Members_Of and Has_Member elements may have arbitrary number of Exclude_Related elements.

Exclude_Related element has Exclude_ID attribute. It is CAPEC identifier, i.e. view, category or attack pattern. The idea of this element is to exclude the ancestor from the relationship (Members_Of and Has_Member).

This mechanism has no sense for views and categories because there is no Members_Of / Has_Member three with path longer than one.

The same element is used in Related_Attack_Patterns element. It is discussed below, but situation there is more confusing.

In reality, CAPEC catalog does not contain Exclude_Related elements for categories and views. Something more, catalog contains only Has_Member elements.

2.2. Categories

Categories element contain Category elements.

Category element has required attributes ID, Name, and Status with the same meaning and purpose as that ones in views.

Summary is required structured text for categories.

Relationships element describes the abstractions hierarchy as Members element for views.

References, Notes, and Content_History have the same structure and meaning as that ones for views.

Taxonomy_Mappings element has Taxonomy_Mapping elements.

Taxonomy_Mapping element has required Taxonomy_Name attribute that can have as value ATTACK, WASC, or OWASP Attack.

Taxonomy_Mapping element subelements are Entry_ID, Entry_Name, and Mapping_Fit. They point to the target taxonomy element and describe mapping effectiveness with the last element that can as value Exact, CAPEC More Abstract, CAPEC More Specific, Imprecise, or Perspective.

2.3. Attack patterns

Attack_Pattern element has required attributes ID, Name, Status, and Abstraction. The last can have as values Meta, Standard, and Detailed.

Meta attacks stay at highest abstraction level. They are defined by specific methodology or attack technique but not by specific technology or implementation.

Meta attacks group standard attacks. They are useful in Architecture/Design phase.

Standard attacks are focused on specific attack technique or methodology. Frequently, they are part of "whole attack".

Standard attacks contain enough details to be understand their specific technique and how achieve its aim.

Detailed attacks are at the lowest abstraction level. They are focused on specific technique or technology. Detailed attack describes the whole execution flow.

Detailed attacks contain protection descriptions against them.

Very frequently, detailed attacks are chains of standard attacks. Therefore, detailed attacks can be placed at several locations in the attack taxonomy.

Description and Extended_Description elements are structured text.

Alternate_Terms element has Alternate_Term elements.

Alternate_Term element has required Term element and optional Description element that is structured text.

Likelihood_Of_Attack element is enumeration of High, Medium, Low, and Unknown values.

Typical_Severity element is enumeration of Very High, High, Medium, Low, and Very Low values.

Execution_Flow element has elements Attack_Step.

Attack_Step has required Step, Phase, Description, and optional Technique elements.

Step element numerates the step.

Phase element is enumeration of Explore, Experiment, and Exploit values.

Technique element is structured text extended with CAPEC_ID attribute. The last one can point to any catalog entry – there is no restriction in CAPEC XSD schema, but has to point to attack technique.

Prerequisites element has Prerequisite elements.

Prerequisite element is a structured text describing conditions for attack success.

Skills_Required element contains Skill elements.

Skill element is string extended with Level attribute that can have value High, Medium, Low, or Unknown.

Resources_Required element has Resource elements that are structured text.

Indicators element contains Indicator elements – structured text. Indicators are activities, events, conditions, or behaviors describing attack preparations, attack under development, or results of successful attack.

Consequences element contains Consequence elements.

Consequence element has Consequence_ID attribute and Scope, Impact, Likelihood, and Note elements.

Scope element is enumeration of Confidentiality, Integrity, Availability, Access Control, Accountability, Authentication, Authorization, Non-Repudiation, and Other.

Impact element is enumeration of Modify Data, Read Data, Unreliable Execution, Resource Consumption, Execute Unauthorized Commands, Gain Privileges, Bypass Protection Mechanism, Hide Activities, Alter Execution Logic, and Other.

Likelihood is enumeration of High, Medium, Low, and Unknown.

Note element is structured text.

Mitigations element has Mitigation elements – structured text describing how to harden the system, to shrink attack surface, or decrease impacts from successful attack.

Example_Instances element has Example elements – structured text.

Related_Weaknesses has Related_Weakness elements.

Related_Weakness element has required attribute CWE_ID – pointing to CWE weakness.

Taxonomy_Mappings, References, Notes and Content_History elements are the same as in categories.

Finally, Related_Attack_Patterns element has Related_Attack_Pattern elements.

Related_Attack_Pattern element has required Nature and CAPEC_ID attributes. CAPEC_ID can point to any catalog entry by CAPEC XSD schema – not only to attack patterns how the name suggests.

Nature attribute is enumeration of ChildOf, ParentOf, CanFollow, CanPrecede, CanAlsoBe, and PeerOf.

Pairs CanFollow/CanPrecede organize unnamed attack patterns.

CanAlsoBe link the attack with a similar one, but inverse relationship is not obligatory.

PeerOf marks relationships that cannot be classified as some of preceding ones.

Pair ChildOf/ParentOf sets relationships between attack patterns at different abstract levels, but there are no such restrictions in CAPEC XSD schema.

Only ChildOf relationship is used in the catalog.

There are only two graph views in the catalog. Meta attacks in them point to standard or detailed attacks.

Related_Attack_Pattern element can contain Exclude_Related elements.

Exclude_Related element has required Exclude_ID attribute that is CAPEC identifier. There are no restrictions in CAPEC XSD schema for Exclude_ID.

Following the logic of relationships ChildOf/ParentOf, only attack patterns participating in this kind of relationships have to be excluded.

CAPEC catalog contains only ChildOf relationships. ChildOf/ParentOf relationship trees start from meta attacks. Therefore, exclusion of an ancestor means to prune the subtrees starting from this ancestor.

An attack pattern can participate in several subtrees of taxonomy hierarchy.

Following this logic, only attack patterns have to be excluded, but in the catalog, only categories are excluded.

Categories and attack patterns can be in Member_Of/ Has_Member relationships. Therefore, ChildOf/ParentOf relationship is extended with Member_Of/ Has_Member relationship.

Something more, there are views (not the basic ones) in the catalog, in which categories directly point to standard attacks without meta attacks.

2.4. External references

External_References elements are sequence of External_Reference elements.

External_Reference element has required Reference_ID attribute and elements Author, Title, Edition, Publication, Publication_Year, Publication_Month, Publication_Day, Publisher, URL, and URL_Date. It points to external detailed information.

3. CAPEC ontology

CAPEC ontology follows CAPEC XSD schema. Description data are annotations in this ontology. Classification data are classes, object, and data properties.

Constructions that have no clear semantic are more difficult for interpretation. In that case, thorough investigation of their real usage in the database has been done. Such examples are attack relationships.

Attack concept presentation is still immature that is why there are many constructs with unclear semantic. Therefore, the best approach is OWL and "open world" assumption for knowledge presentation.

CAPEC ontology includes as annotations some referent information: schema, date, copyright, dictionary name, version, version date, ontology author, and ontology version.

CAPEC ontology contains external references. They are presented as External_Reference annotations that are literals structured by Author:, Title:, Edition:, Publication year:, Publication month:, Publication day:, Publisher:, URL:, URL date: and Reference_ID: – subelements of External_Reference element.

Figure 1 shows CAPEC ontology common class schema.

Figure 2 shows CAPEC class hierarchy.

3.1. Views

CAPEC ontology is a set of views. View class presents view concept. This class is union without intersection of its subclasses Explicit, Graph, and Implicit.

Type data property presents view type and can has as value one of the abovementioned subclasses names. This is some unnecessary information duplication but for CAPEC XSD schema "compatibility" is available.

Figure 1. CAPEC ontology common class schema

Explicit class has no instances at all in the catalog. There are some explicit views defined as implicit ones – query of these implicit views simply lists class members.

There are two main graph views. They have strong hierarchical structure view – category – meta attack – standard attack – detailed attack. However, there are some deviations from this structure in other view – for example, some meta attacks may have as children detailed attacks.

Query defines implicit views. Reasoners do not execute queries. That is why implicit view content is deployed with Has_Member object property pointing to attack from query execution result.

```
Class: View
    DisjointUnionOf:
        Explicit, Graph, Implicit
Class: Explicit
    SubClassOf:
```

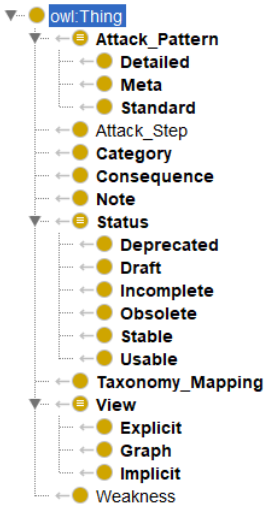

Figure 2. CAPEC class hierarchy

```
        View
Class: Graph
    SubClassOf:
        View
Class: Implicit
    SubClassOf:
        View
```

Annotation definitions will be not presented here.

View status is presented by Status class (categories and attacks have status too). Status class is union without intersection of views, categories, and attacks.

On the other hand, Status class has subclasses Deprecated, Draft, Incomplete, Obsolete, Stable, and Usable. Status class is union without intersection of these subclasses too.

Status class is not "abstract class" – class without instances. If an individual is from only Status class this means that still is not clear to which of Status classes this individual belongs.

Abstract classes can be controlled via SHACL [3] because Semantic Web uses "open world" assumption.

Above considerations are true for views.

```
Class: Status
    DisjointUnionOf:
        Attack_Pattern, Category, View
    DisjointUnionOf:
        Deprecated, Draft, Incomplete, Obsolete, Stable,
```

```
        Usable
Class: Deprecated
    SubClassOf:
        Status
Class: Draft
    SubClassOf:
        Status
Class: Incomplete
    SubClassOf:
        Status
Class: Obsolete
    SubClassOf:
        Status
Class: Stable
    SubClassOf:
        Status
Class: Usable
    SubClassOf:
        Status
```

Inverse pair of object properties Has_Member/Member_Of presents membership of categories and attacks to view. CAPEC XSD schema does not control in any way this membership. It is possible some view to be member of another view.

Total control on memberships in the ontology can be achieved via SHACL.

Above consideration are applicable for category membership.

Following the membership relationships, Has_Member object property has as domain views and categories, and as range views, categories, and attacks.

```
ObjectProperty: Has_Member
    Domain:
        Category or View
    Range:
        Attack_Pattern or Category or View
    InverseOf:
        Member_Of
ObjectProperty: Member_Of
    Domain:
        Attack_Pattern or Category or View
    Range:
        Category or View
    InverseOf:
        Has_Member
```

Objective element is presented as annotation.

View has Audience data property of enumeration type of values Academic Researchers, Applied Researchers, Assessment Customers, Assessment Vendors, CAPEC

Team, Educators, Information Providers, Other, Software Customers, Software Designers, Software Developers, and Software Vendors.

```
DataProperty: Audience
    Domain:
        View
    Range:
        {"Academic Researchers",
         "Applied Researchers",
         "Assessment Customers", "Assessment Vendors",
         "CAPEC Team", "Educators", "Information Providers",
         "Other", "Software Customers",
         "Software Designers", "Software Developers",
         "Software Vendors"}
```

Audience data property can be accompanied by Audience_Description annotation.

Audience data property and Audience_Description annotation come from Stakeholder element. The last is not used in the ontology.

Only implicit views have Filter annotation. It contains XPath query from which view content is deployed via Has_Member object property.

Reference annotation presents view references – an annotation for a reference. Here, External_Reference_ID and Section elements are embedded on separate lines.

View notes have structure that is more informative. They are deployed via Note class and Note object property pointing to individuals of that class. Notes have Type data property and Note_Description annotation. Note's Type can have as value Maintenance, Relationship, Research Gap, Terminology, or Other.

```
ObjectProperty: Note
    Domain:
        Attack_Pattern or Category or View
    Range:
        Note
Class: Note
DataProperty: Type
    Characteristics:
        Functional
    Domain:
        Note
    Range:
        {"Maintenance", "Other", "Relationship",
         "Research Gap", "Terminology"}
```

Current_History annotation presents the catalog element with the same name. Its subelements (Submission, Modification, Contribution, and Previous_Entry_Name) are embedded in Current_History on separate lines with offsets.

References, notes, and content history are available for categories and attacks with the same structure, interpretation, and meaning.

View individual IRI is constructed with view ID attribute. The last is ID number prefixed with "CAPEC_". Category and attack individuals construct their individual IRIs in the same way.

```
DataProperty: ID
    Characteristics:
        Functional
    Domain:
        Attack_Pattern or Category or View
    Range:
        xsd:positiveInteger
```

View name is presented with Name data property. Category and attack names are presented with the same data property.

```
DataProperty: Name
    Characteristics:
        Functional
    Domain:
        Attack_Pattern or Category or View
    Range:
        xsd:string
```

3.2. Categories

Category class presents categories. Category groups attacks on particular target or effect.

```
Class: Category
```

Category description aim is given with Summary annotation.

Category members are linked with its category via Member_Of object property (inverse object property is Has_Member). These object properties are described in view subsection.

Categories can be mapped to other taxonomies – not only which are presented in CAPEC. This mapping is more informative. Taxonomy_Mapping class, and object property with the same name are used for that purpose. The object property points to individuals from the class.

```
Class: Taxonomy\_Mapping
```

Taxonomy_Mapping class has Taxonomy_Name data property. The last may have as value ATTACK, WASC or OWASP Attacks. Taxonomy_name is required but this can be controlled with SHACL.

```
DataProperty: TaxonomyName
    Characteristics:
        Functional
    Domain:
        Taxonomy_Mapping
    Range:
        {"ATTACK", "OWASP Attacks", "WASC"}
```

Entry_ID, Entry_Name, and Mapping_Fit fix the mapping. First two sets target taxonomy name and entry name in this taxonomy. Mapping_Fit describes how precise is the mapping: Exact, CAPEC More Abstract, CAPEC More Specific, Imprecise, or Perspective.

```
DataProperty: Entry_ID
    Characteristics:
        Functional
    Domain:
        Taxonomy_Mapping
    Range:
        xsd:string
DataProperty: Entry_Name
    Characteristics:
        Functional
    Domain:
        Taxonomy_Mapping
    Range:
        xsd:string
DataProperty: Mapping_Fit
    Characteristics:
        Functional
    Domain:
        Taxonomy_Mapping
    Range:
        {"CAPEC More Abstract", "CAPEC More Specific",
         "Exact", "Imprecise", "Perspective"}
```

3.3. Attack patterns

Attack pattern by its abstraction level can be meta, standard, or detailed. Attack_Pattern class and its subclasses Meta, Standard, and Detailed present attack patterns and their abstraction levels.

Attack_Pattern class is union without intersection of its subclasses.

Attack_Pattern class can be viewed as "abstract class" in the "closed world".

```
Class: Attack_Pattern
    DisjointUnionOf:
```

```
        Detailed, Meta, Standard
Class: Detailed
    SubClassOf:
        Attack_Pattern
Class: Meta
    SubClassOf:
        Attack_Pattern
Class: Standard
    SubClassOf:
        Attack_Pattern
```

References, notes, and content history are as in categories.

Attack_Pattern IRIs, statuses, and attack names are as in categories.

Attack_Pattern_Description annotation describes the attack. Required annotation eventually can be supported by SHACL.

Extended_Description annotation can be attached to attacks.

Alternative attack names are presented with Alternate_Term data property and its annotation Alternate_Term_Description.

```
DataProperty: Alternate_Term
    Domain:
        Attack_Pattern
    Range:
        xsd:string
```

Likelihood_Of_Attack is functional data property that can have one of High, Medium, Low, or Unknown.

```
DataProperty: Likelihood_Of_Attack
    Characteristics:
        Functional
    Domain:
        Attack_Pattern
    Range:
        {"High" , "Low" , "Medium" , "Unknown"}
```

Typical_Severity is functional data property that can have one of Very High, High, Medium, Low, or Very Low.

```
DataProperty: Typical_Severity
    Characteristics:
        Functional
    Domain:
        Attack_Pattern
    Range:
        {"High" , "Low" , "Medium" , "Very High" ,
         "Very Low"}
```

Execution_Flow object property points to individuals of Attack_Step class. Attack_Step individual IRI is constructed by attack's IRI suffixed with "_Attack_Step" and step number starting from zero.

```
ObjectProperty: Execution_Flow
    Domain:
        Attack_Pattern
    Range:
        Attack_Step
Class: Attack\_Step
```

Attack_Step class has required functional property Step. It is step sequence number. Numbering starts with one and has no relations with IRI individual construction for the class.

```
DataProperty: Step
    Characteristics:
        Functional
    Domain:
        Attack_Step
    Range:
        xsd:positiveInteger
```

Phase is required functional property of Attack_Step. It can have as value one of Explore, Experiment, and Exploit – attack phase in which the step is executed.

```
DataProperty: Phase
    Characteristics:
        Functional
    Domain:
        Attack_Step
    Range:
        {"Experiment" , "Exploit" , "Explore"}
```

Attack_Step individuals must be annotated with Attack_Step_Description.

Technique applicable in the attack step can optionally be another attack pattern. This option can be implemented via Technique object property pointing to another attack pattern. Nevertheless, CAPEC XSD schema permits this element to point to views and categories.

```
ObjectProperty: Technique
    Domain:
        Attack_Step
    Range:
        Attack_Pattern
```

If attack pattern is attached to the step then the object property may be annotated with Technique_Description. Otherwise, this annotation is applied to the step individual. Therefore, technique description can appear in two different locations. The idea may be is that in thoroughly analyzed systems, attack steps must be attack patterns.

Technique description appears in Technique element whose content is structured text optionally extended with CAPEC_ID attribute. The last sets the value of Technique object property.

```
ObjectProperty: CAPEC_ID
    Characteristics:
        Functional
    Range:
        Attack_Pattern
```

Prerequisite annotation presents preliminary requirements for attack success.

Required attacker skills for attack execution are presented with Skill data property that can have as value one of High, Medium, Low, or Unknown. This property can be annotated with Skill_Description.

```
DataProperty: Skill
    Domain:
        Attack_Pattern
    Range:
        {"High", "Low", "Medium", "Unknown"}
```

Required resources for successful attack are defined with Resource annotations.

Indicator annotations describe attack preparation, ongoing attack or finished attack indicators.

Attack consequences are described via Consequence object property pointing to individuals of class with the same name. Individual IRI of that class are constructed by attack IRI suffixed with "_Consequence" and consequence successive number starting from zero.

```
ObjectProperty: Consequence
    Domain:
        Attack_Pattern
    Range:
        Consequence
Class: Consequence
```

Consequence class has Scope data property with possible values Confidentiality, Integrity, Availability, Access Control, Accountability, Authentication, Authorization, Non-Repudiation, and Other, representing security areas impacted by successful attack execution. At least one security area must be marked.

```
DataProperty: Scope
    Domain:
        Consequence
    Range:
        {"Access Control", "Accountability",
         "Authentication", "Authorization", "Availability",
         "Confidentiality", "Integrity", "Non-Repudiation",
         "Other"}
```

Another data property of Consequence class is Impact with possible values Modify Data, Read Data, Unreliable Execution, Resource Consumption, Execute Unauthorized Commands, Gain Privileges, Bypass Protection Mechanism, Hide Activities, Alter Execution Logic, or Other. These are technical consequences of successful attack.

```
DataProperty: Impact
    Domain:
        Consequence
    Range:
        {"Alter Execution Logic",
         "Bypass Protection Mechanism",
         "Execute Unauthorized Commands", "Gain Privileges",
         "Hide Activities", "Modify Data", "Other",
         "Read Data", "Resource Consumption",
         "Unreliable Execution"}
```

Consequence likelihood is given by Likelihood data property, which is functional and can have as value one of High, Medium, Low, or Unknown.

```
DataProperty: Likelihood
    Characteristics:
        Functional
    Domain:
        Consequence
    Range:
        {"High", "Low", "Medium", "Unknown"}
```

Consequence may be attached with Consequence_ID identifier (for internal use) and Note annotation.

```
DataProperty: Consequence_ID
    Characteristics:
        Functional
    Domain:
        Consequence
    Range:
        xsd:string
```

Attack mitigations (preventions) are not formalized and are presented with Mitigation annotations.

Attack examples are not formalized too and are presented with Example annotations.

Weaknesses that are exploited by the attack are referred via Related_Weakness object property. There are no comments on weakness role. The property simply point to an individual from CWE ontology.

```
ObjectProperty: Related_Weakness
    Domain:
        Attack_Pattern
    Range:
        cwe:Weakness
```

Finally, related attack patterns are discussed. They generically are presented with Related_Attack_Pattern object property. This property has to be "abstract" in the "closed world". "Abstract" classes have no instances, so "abstract" properties have no links (instances).

```
ObjectProperty: Related_Attack_Pattern
    Domain:
        Attack_Pattern
    Range:
        Attack_Pattern
```

Related_Attack_Pattern has subproperties ChildOf, ParentOf, CanFollow, CanPrecede, CanAlsoBe, and PeerOf. Each of them presents some kind of relationship.

Inverse pairs are ChildOf/ParentOf and CanFollow/CanPrecede.

```
ObjectProperty: ChildOf
    SubPropertyOf:
        Related_Attack_Pattern
    InverseOf:
        ParentOf
ObjectProperty: ParentOf
    SubPropertyOf:
        Related_Attack_Pattern
    InverseOf:
        ChildOf
ObjectProperty: CanFollow
    SubPropertyOf:
        Related_Attack_Pattern
    InverseOf:
        CanPrecede
ObjectProperty: CanPrecede
```

```
    SubPropertyOf:
        Related_Attack_Pattern
    InverseOf:
        CanFollow
ObjectProperty: CanAlsoBe
    SubPropertyOf:
        Related_Attack_Pattern
ObjectProperty: PeerOf
    SubPropertyOf:
        Related_Attack_Pattern
```

Inverse relationships form chains of relationships. ChildOf/ParentOf organize abstraction hierarchy. ChildOf, must link in principle the attack with another attack from the same or at higher abstraction level. However, there is no such restriction in CAPEC XSD schema.

De facto, CAPEC XSD schema does not put any restrictions on CAPEC_ID attribute of Related_Attack_Pattern element. It is possible this attribute to point even to view or category.

Obviously, authors of CAPEC XSD schema have not fixed above-mentioned problem and Related_Attack_Pattern intend has to be detected from the database content. It is possible in future the problem to be fixed and even sustainable changes to CAPEC XSD schema to be done.

In CAPEC database content, Related_Attack_Pattern points only to attack patterns. It is possible to use SHACL for more strict control on these relationships, for example, ChildOf to point to attack patterns on the same or higher abstraction level.

Pair CanFollow/CanPrecede may organize chains of unnamed attack patterns. Attack chain concept is similar to that one in CWE but not so advanced.

CanAlsoBe and PeerOf do not organize relationship chains. CanAlsoBe relates pair of attacks but this relationship is not commutative or transitive. PeerOf relates two attacks in relationship that cannot be classified as relationship from the other five listed kinds.

Related_Attack_Pattern property has for domain only attack patterns.

From analysis of database's content follows that follows that only categories are excluded from relationship trees. Therefore, exclusion is not applied as universally as defined by CAPEC XSD schema. In conclusion, abstraction subtrees organized by ChildOf/ParentOf and Member_Of/Has_Member are pruned only on category level.

It is desirable CAPEC XSD schema to be changed following the real construction usage. Even more, many more upgrades of this schema are expected in the future. For now, CAPEC ontology follows current CAPEC XSD schema and some restrictions are applied considering CAPEC database content.

Now ancestor exclusion will be discussed. From content of CAPEC database follows that only category ancestors are excluded. Between excluded category and

the attack under consideration there are zero or more attack linked with that category. The last can be at any abstraction level by CAPEC XSD schema, but in practice, levels are meta and standard. Excluded category is linked with attack under consideration via Member_Of/Has_Member and zero or more ChildOf/ParentOf relationships. Ancestor tree starts from view, then pass to categories via Member_Of/Has_Member, and after that via zero or more ChildOf / ParentOf ends in the attack under consideration.

Category children define category of its descendants in ancestor tree – some kind of category type. Therefore, via Exclude_Related elements only specific categories are excluded as ancestors for the attack under consideration. It follows that independent of the fact that there is a path organized by Member_Of/Has_Member and ChildOf/ParentOf, only categories are excluded. As result of these considerations Exclude_Related object property excludes only categories. May be in the future this restriction will be changed – maybe not, or CAPEC taxonomy or CAPEC XSD schema will be changed.

4. Conclusion

CAPEC schema analysis and especially analysis on relationships shows that CAPEC is still not mature. Therefore, CAPEC schema upgrades will be developed in the future.

In this situation, is there any sense to formalize CAPEC knowledge in OWL? Concept of "closed world" supposes that all knowledge is collected in knowledge base. This means that statements not following from this knowledge base are not true.

In "open world" concept if something is not known may be true.

In mature taxonomy, the database contains all knowledge. This means that when CAPEC get mature, its database would cover "closed world" criteria. Is it true?

Currently, there are very many questions about some attacks, i.e. this problem area is under intensive investigations and is way from "closed world".

On the other hand is the future. Eventually, CAPEC taxonomy will get stable enough to classify it as mature. The problem to happen this is that new complicated unknown attacks are under development. Therefore, there is no way CAPEC taxonomy to get mature. In this situation more applicable is "open world" concept, because CAPEC taxonomy is unstable – under permanent development.

What is NIST approach to CVE? NIST extracts from CVE only analyzed vulnerabilities for NVD. In NVD, NIST guarantees the "closed world" concept.

Why CPE, NVD, CVE, CWE, and CAPEC are not presented as ontologies? All these databases are under permanent development. If these databases were presented as ontologies, there would be no need to do transformations like CVE to NVD. Something more, reasoners can open new horizons for research and investigations. Finally, "closed world" aspects can be checked with SHACL.

Acknowledgements

This paper is prepared with the support of MIRACle: Mechatronics, Innovation, Robotics, Automation, Clean Technologies – Establishment and development of a Center for Competence in Mechatronics and Clean Technologies – Laboratory Intelligent Urban Environment, funded by the Operational Program Science and Education for Smart Growth 2014–2020, Project BG 05M2OP001-1.002-0011.

References

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns: Elements of reusable object-oriented software, Boston, MA: Addison-Wesley, 1995.

[2] MITRE Corporation, CAPEC (Common Pattern Enumeration and Classification), 2023. URL: http://capec.mitre.org.

[3] W3C, Shapes Constraint Language (SHACL), W3C Recommendation 20 July 2017. URL: http://www.w3.org/TR/shacl.

Received on March 20, 2023
Accepted on May 7, 2023

Vladimir Dimitrov

Faculty of Mathematics and Informatics
Sofia University "St. Kliment Ohridski"
5 James Bourchier Blvd.
1164 Sofia
BULGARIA

E-mail: cht@fmi.uni-sofia.bg