CPE Ontology

Vladimir Dimitrov [0000-0002-7441-253X]

Sofia University „St. Kliment Ohridski", Faculty of Mathematics and Informatics
1164 Sofia, 1 James Bourchier Blvd., Bulgaria

cht@fmi.uni-sofia.bg

Abstract. Common Platform Enumeration (CPE) is maintained by NIST as a structured schema for naming of information technology systems, software and packages. CPE is described in a several NIST documents (the last version):

- Common Platform Enumeration: Naming Specification. Version 2.3.
- Common Platform Enumeration: Name Matching Specification. Version 2.3.
- Common Platform Enumeration: Dictionary Specification. Version 2.3
- Common Platform Enumeration: Application Language Specification. Version 2.3.

CPE names are building block in NIST classification systems for vulnerabilities (CVE), weaknesses (CWE) and attack patterns (CAPEC).
In this paper, CPE names are described as an ontology in OWL Manchester Syntax. This ontology is generated from NIST "Official Common Platform Enumeration Dictionary". Its purpose is to be a reference ontology for the ontologies representing vulnerabilities, weaknesses and attack patterns.

Keywords: CPE, OWL, Ontology, Cybersecurity.

1 Introduction

CPE (Common Platform Enumeration) has been developed by MITRE Corporation [1], but since November 2014, it is supported only by NIST [2]. This is a structured naming scheme for information technology systems, software and packages (platforms).

The latest version of the CPE is 2.3 and is described in the NIST series of publications [3-6].

The naming specification is presented in [3]. Here, the logical structure of CPE names is defined.

The name matching specification [4] describes a method for comparing two CPE names. Here, CPE names are treated as search patterns. In general, there is no difference in the notation for CPE names and search patterns.

The official NIST dictionary [5] contains only "basic" CPE names. These "basic" CPE names can also be considered as search patterns because they rep-

resent a class of platforms that have the same vulnerabilities, vulnerabilities, and attack patterns.

Each search pattern corresponds to a set of "basic" CPE names in the dictionary. The "basic" CPE name as a search pattern has only one element – the "basic" CPE name itself.

The publication [4] defines how CPE names are compared as search patterns, i.e. through the sets of their corresponding "basic" CPE names.

Complex CPE name configurations can be specified with the CPE applicability language [6]. These configurations are used in guidelines, policies, and other places where platforms are referenced as CPE names.

CPE names are the main building blocks in the NIST specifications and in the maintained vulnerability databases [7].

In NVD, CPE names are used to describe platforms that are affected by the vulnerabilities. This is done through the applicability language. Statements in this language refer to vulnerable platform configurations.

Before proceeding to the CPE ontology, a brief presentation the naming specification has to be done.

2 CPE naming

The CPE names are described with an abstract naming model WFN (Well-Formed Name). Abstract names can be bounded to specific syntax. There are two specific syntaxes described in [3]: URI [8] and FS (Formatted String).

WFN is just a logical construction in this specification. It is an unordered set of attribute-value pairs that are in the form "attribute = value".

Attribute names are not case sensitive. The underline (the symbol "_") is treated as a letter.

The WFN name format is:

```
wfn:[a1=v1, a2=v2, …, an=vn]
```

The following attribute names are used: "part", "vendor", "product", "version", "update", "edition", "language", "sw_edition", "target_sw", "target_hw" and "other". Each attribute can participate no more than once in the CPE name. If an attribute does not participate in the CPE name, its value by default is the logical value ANY.

Attribute values can be "logical" or character strings. Additional restrictions may be imposed on the individual attributes.

The "logical" values are ANY and NA. The first value ANY (any value) means that any value is applicable for that attribute. The second value NA (not applicable / not used) means that for this attribute cannot be assigned any meaningful or valid value, or that the attribute is not used in the name.

The attribute values that are character strings consist of printed UTF-8 characters. They are enclosed in quotation marks.

The underscore is considered as a letter.

There are three special characters: "\" backslash, "*" star and "?" question mark.

The backslash is used to escape characters. All non-alphanumeric characters must be escaped in the attribute value string, including the backslash itself when used without its special meaning. The attribute value string cannot be "\ _".

The star means zero or more random symbols in place, and the question mark – one random symbol. In fact, these two symbols are used to set matching patterns. However, there are restrictions on their use:

- The special characters "*" and "?" can be at the beginning or end, i.e. cannot be used in the middle of the pattern string.
- A single star cannot be an attribute value.
- The star cannot be used more than once in a sequence.

The question mark can be a single attribute value, and can be used more than once in a row.

Star and a question mark at the beginning or the end of the string does not make sense, but it is permitted the string to starts or ends with a star and several question marks.

These rules for attribute values are presented in [3] as a grammar in ABNF (Augmented Backus-Naur Form).

Specific restrictions on attribute values are:

- The "part" attribute can have a value of "a" for applications; "o" for operating systems; or "h" for hardware devices.
- The "edition" attribute, in general, has ANY value, but may have a string value for backward compatibility with the previous versions of the specification.

The names of the attributes correspond to their meaning, but there are attributes whose names are not so directly clear:

- "sw_edition" fixes the product to a specific market or class of end users.
- "target_sw" determines the software environment in which the product operates.
- "target_hw" sets the architectural set of machine instructions. Bytecode intermediate languages are also considered as an architectural set of machine instructions.
- "language" is a valid language label according to RFC 5646 [10], but only language and region codes are used. This attribute describes the language of the user interface.
- "other" is anything else that can be used to identify the class.

Examples from [3] for WFN names:

```
wfn:[part="a", vendor="microsoft", product="internet_
explorer", version="8\.0\.6001", update="beta",
edition=NA]

wfn:[part="a", vendor="hp", product="insight_
diagnostics", version="7\.4\.0\.1570", sw_
edition="online", target_sw="windows_2003", target_
hw="x64"]
```

There are three operations defined for WFN:

- `new()` – creates an empty WFN, i.e. there are no attribute-value pairs in it, or, equivalently, the attributes are initialized with ANY value.
- `get(w, a)` – from WFN (argument "w") returns the value of the second argument ("a"). If no value is set for this attribute, it will return ANY.
- `set(w, a, v)` – in the WFN "w" of the attribute "a" sets the value "v".

The absence of an attribute in the WFN is equivalent to initializing it with an ANY value, and then deleting an attribute is equivalent to assigning to it the ANY value.

The above considerations have been used in the implementation of a module for manipulating CPE names. The original idea was to use the cpe 1.2.1 module from the Python repository. This module supports all published versions of CPE: 1.1, 2.2 and 2.3, as well as the syntax of WFN, URI and FS. The module has minimal gaps in its implementation, but in general, it turned out to be completely unusable for processing huge amounts CPE names – it is too slow. Most likely this is due to the universal and simultaneous support of the all three versions of the specification.

The above limitations necessitated the development of new module only for CPE 2.3 and with minimum functionality required only for conversion between the three formats (WFN, URI and FS).

The binding of the abstract WFN is done with the two specific formats URI and FS. In the specification, the URI format is included rather for backward compatibility with previous versions, while preference is given to the FS format.

From the point of view of OWL ontologies, an URI is more appropriate for identifier of individuals in an ontology because it is an IRI in the sense of an OWL – each URI is an IRI.

FS simply structures attributes without considering their representation capabilities as IRI. In fact, the content of an FS can be considered as a set of attribute values in the CPE ontology linked as values to data properties. Therefore, it is unnecessary to maintain FS in the ontology in any other way.

3 Binding WFN to URI

The URI syntax used in CPE is presented in [3] as a grammar with ABNF. This is a restriction on the URI.

The language label is in a simplified version.

Note to the case: It is generally assumed that the control of CPE contents is performed by NIST, i.e. the language label is set according to the requirements. New CPEs are not entered in the ontology; instead, they are imported from the CPE dictionary.

The CPE's URI starts with "cpe:/" (header or prefix) and then follow the attribute's values of "part", "vendor", "product", "version", "update", "edition", and "language" in the specified order and separated by a colon. There is also a colon after the header and before "part". If a value of an attribute is missing, its place is marked with an empty string, i.e. "::".

The absence of trailing attributes until the header itself is allowed. Simply, the last missing values and their colons are omitted. |The URI's format with removed trailing empty attribute values will be called "compressed" URI format.

CPE URI version 2.3 maintains backward compatibility with CPE URI version 2.2. This is achieved by "packaging" the new attribute values with the value of the "edition" attribute. It is discussed below.

Special characters and punctuation, in the attribute values, coding with the percentage notation is required.

The logical value ANY is represented in the URI with the empty string at the position of the attribute, i.e. "" or in place of the attribute position it is represented as "::".

The logical value NA is represented by "-".

In WFN, non-alphanumeric characters in attribute value strings must be preceded by "\", except for "*" and "?" when they have been used as special symbols.

In URI, non-alphanumeric characters are represented in percentage encoding.

In WFN, the characters minus "-" and dot "." are "escaped", i.e. they are represented as "\-" and "\.". In URI, they are not in the percentage notation.

The special characters "?" and "*" when not "escaped" are represented by "%01" and "%02" encodings, respectively.

3.1. Packaging of the new attributes

CPE version 2.3 has four new attributes: "sw_edition", "target_sw", "target_hw", and "other". When one of them has a value other than ANY, they are "packaged" in the specified order in the component "edition" of the URI. Otherwise they are skipped. In the packaged format, the symbol "~" is used as a separator. In fact,

the sixth component of the URI packs five attributes: the old “edition” and the four new attributes. Packaging consists of concatenation of attribute values in the specified order, using “~” as a prefix and delimiter.

If the four new attributes have an ANI value, then no packaging is performed and the only value in the sixth component is that of “edition”.

4 Unbinding URI to WFN

If the URI meets the format requirements of CPE 2.3, then it can be “unbind” to WFN.

URI components 1-5 and 7 are decoded in the corresponding values as:

- the empty string is represented by the logical value ANY;
- the unit value “-” – with NA;
- the dot “.” and minus “-” are escaped;
- and decode the percentage forms.

The sixth component is unpacked if it starts with “~”.

5 Binding WFN to FS

The formatted string is similar to URI, but unfortunately, it is not. Many of URI restrictions are released.

The formatted string starts with “cpe:2.3:” and is followed by the attribute values separated by “:”. Its format is:

```
cpe:2.3: part : vendor : product : version : update :
edition : language : sw_edition : target_sw : target_hw :
other
```

The symbols that are not “escaped” in the values of its attributes are the letters, numbers, “-”, “.” and “_”.

The logical values ANY and NA are represented by “*” and “-”, respectively.

All other characters are “escaped” with “\”. The special symbols for the logical values, when used as such (as a stand-alone value), are not “escaped”.

Each attribute must have a value in the formatted string.

The binding process is relatively simple. Each WFN attribute value is processed separately and then the resulting processed values are concatenated. Since there is a difference between the “escaped” symbols in WFN and FS, the work must be a bit more precise here. Each “escaped” symbol is checked (if preceded by “\”). The characters “.”, “-” and “_” are “escaped” in WFN, but not in FS. The conversion of “escaped” characters in WFN to characters in the formatted string is:

- When a character is “escaped” in WFN, it remains the same in FS.
- From the “escaped” dot, minus and underscore (“.”, “-” and “_”) is removed the escape symbol (“\”).

The binding and unbinding algorithms are described in pseudo code in the `bind_to_fs(wfn)` and `unbind_to_fs(uri)` functions.

There are two other functions recommended by the specification: `convert_uri_to_fs(uri)` and `convert_fs_to_uri(fs)`.

6 CPE ontology

The ontology has been developed in OWL 2 Manchester Syntax [11]. The reason for this is that Manchester Syntax is compact and easy to read by humans.

The development was carried out with the Protégé tool [12], which supports the above version of OWL.

A general scheme of the classes is presented in Fig. 1.

The ontology follows the specification of CPE 2.3, but it is an incomplete implementation of CPE 2.3 requirements and recommendations. It cannot be regarded as CPE 2.3 compatible implementation.

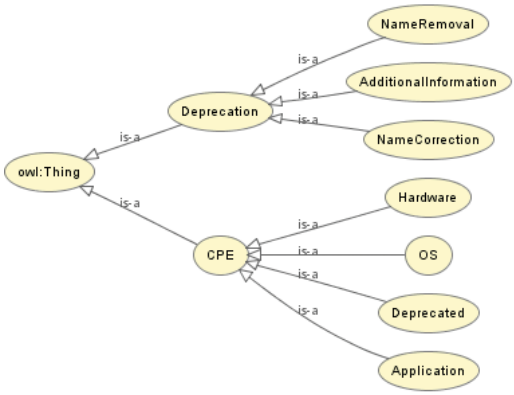

Fig. 1. Class hierarchy.

URI identifiers are selected to be ontology individual identifiers. Each CPE name description in the NIST dictionary contains both a FS name and an URI name. The latter is suitable for IRI individuals in the ontology.

CPE is the main class in the ontology. It has three main subclasses: Application, Hardware and OS, which correspond to the classification given by the "part" attribute. In such a way, greater search flexibility is achieved and the "part" attribute itself does not need to be stored as a data property.

The class Deprecated is the fourth subclass of CPE. This is the subclass of obsolete names. An obsolete name can be from the other three subclasses of CPE and therefore Deprecated is a subclass of one of Application, Hardware and OS. The definition of this class is:

```
CPE and (Application or Hardware or OS)
```

The class Deprecation is at the level of the class CPE. This utility class associates the obsolete name with its replacements. There are several possible reasons a name to be deprecated and for each of them a subclass has been defined: AdditionalInformation, NameCorrection and NameRemoval.

It is not clear to what extent obsolete names would play role into the practical use of the ontology.

The hierarchy of classes is presented partly in Fig. 2.

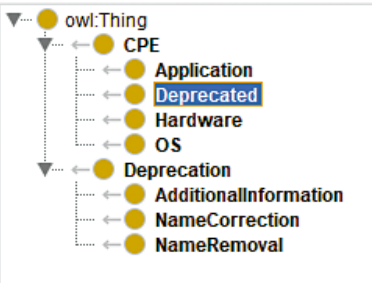

Fig. 2. Another view on class hierarchy.

Deprecated name is associated with an individual of class Deprecation. This is done through the object property "deprecation" of class Deprecated. In this individual (of class Deprecation), the reason for cancellation of the name by subclasses is specified. In the current version of the dictionary, there are mainly deprecated names that have been changed, i.e. of class NameCorrection. Maybe over time the situation will change and other reasons for canceling the names will appear. The definition of the object property "deprecation" is presented below:

```
ObjectProperty: deprecation
    Domain:
        Deprecated
    Range:
        Deprecation
```

Correcting one name or adding additional information about it may result in several new names. Therefore, an individual in the subclasses of Deprecation may point to several new (or already deprected) names. The object property "deprecated-by" of the class Deprecation is used for this purpose. The following is the definition of "deprecated-by":

```
ObjectProperty: deprecated-by
    Domain:
        AdditionalInformation or NameCorrection
    Range:
        CPE
```

When a name is removed, then there should be no object property "deprecated-by" in the corresponding individual from class NameRemoval.

The considerations in the above two paragraphs should be reflected in special axioms, but this is not done for reasons of possible errors in CPE dictionary, which would block loading of the entire ontology in Protégé.

Every change of name (its cancellation) took place on a certain date. A name may be deprecated gradually, i.e. may be replaced by several new names at different times. The name can be changed or even removed. Obviously, the consideration a removed name not to be discarded (even when it is revoked), is to support the systems that use dictionaries with older content. For example, a CVE weakness that references a deprecated name continues to refer to it without the need to change the reference to it in the CVE. Most likely, the vulnerability itself will disappear at some point, but at least until the deprecated CPE name is referenced, it should be kept in the dictionary. However, the CPE dictionary is primarily intended to serve other systems.

In addition to this scheme of aging CPE names, version 2.3 also supports compatibility with previous versions of the specification. In the latter case, one CPE name is replaced by no more than one other CPE name (removed ones are not replaced). For this purpose it uses the object property "deprecated_by" (here is an underscore, not a minus) of class Deprecated.

Most likely, the outdated name substitution mechanism (before CPE 2.3) will not be used, at least in the current content of the dictionary. This will require the omission of the object property "deprecated_by" with all possible axioms for its maintenance. However, an axiom requiring the presence of an object property "deprecation" in individuals of Deprecated has to be added. In the current version of the ontology, neither is done, because backward compatibility is supported. The following is the definition of "deprecated_by":

```
ObjectProperty: deprecated_by
    Characteristics:
        Functional
    Domain:
        Deprecated
    Range:
        CPE
```

Deprecated names can be viewed as a tree whose lists are removed CPE names or current ones.

There are two problematic attributes in the old and the new version of the specification. These are the "deprecation-date" and "deprecation_date" data properties. The last property is inherited from the previous version of the specification and it indicates the date of deprection of the CPE name. This property is a data

property in Deprecated. The following are the definitions of “deprecation-date” and “deprecation_date”:

```
DataProperty: deprecation-date
    Characteristics:
        Functional
    Domain:
        Deprecation
    Range:
        xsd:dateTime
DataProperty: deprecation_date
    Characteristics:
        Functional
    Domain:
        Deprecated
    Range:
        xsd:dateTime
```

On the other hand, according to the new scheme, a CPE name can be deprecated in stages and therefore each stage is reflected with an individual from Deprecation where the “deprecation-date” property is used.

Both data properties are used in the dictionary, but the relationship between them is not obvious and at least such an information have not been found.

The other data properties used are the WFN attributes (without “part”). They belong to the class CPE. The values of these attributes are according to FS rules. There is no data property for the FS identifier. The definitions of the data properties are uniform. Fig. 3 presents the definition of data property “product”.

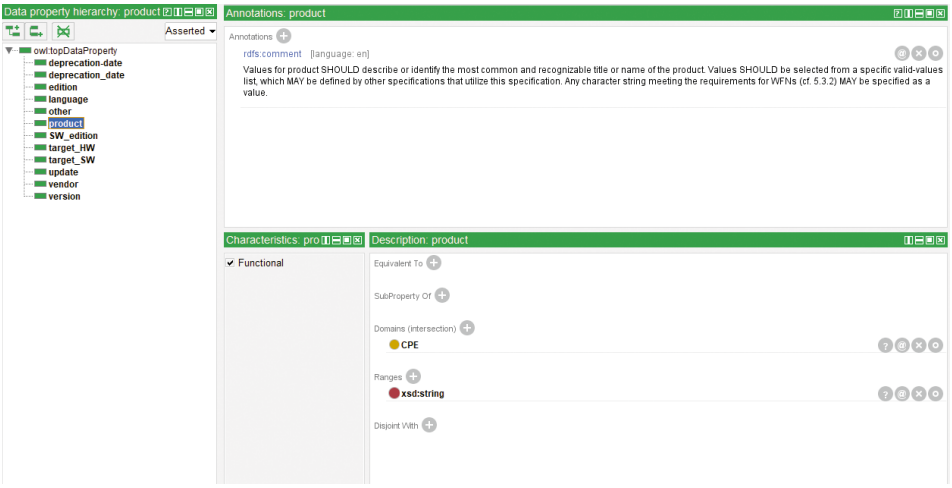

Fig. 3. Definition of data property “product”.

The data property “language” is simply a character string – it does not need to follow RFC 5646. The value is controlled in the NIST dictionary.

Finally, in addition to the dictionary specification, there are a number of descriptions related to official information on the maintenance of the dictionary. This information is transferred to the ontology annotation system.

It turned out that, at least officially, the official information is not maintained in NIST. However, its description is included insofar as it is available in the specification.

7 Conclusion

The CPE ontology complies with the CPE 2.3 specification. It includes its requirements for backward compatibility with the previous versions.

In the ontology development, the basic concept is the minimalism but complete inclusion of dictionary information. This means that information, which does not actually participate in the dictionary or is descriptive or duplicated, is presented in annotations or ignored. In this context, preference is given to removing attributes at the expense of building a hierarchy of classes.

The second concept in ontology design is maximum search flexibility with SPARQL. This implies the elimination of possible additional no-SPARQL search. For example, searching CPE patterns involves searching with regular expressions. The latter mechanism can be used to achieve full compatibility with the requirements for CPE implementation, but is inadequate with the concepts of OWL. The ontology is an extended environment used in conjunction with other ontologies and inference engines.

The ontology, and more precisely its content (individuals), is generated from NIST official CPE dictionary. At this time, NIST does not plan to shift CPE dictionary from XML to OWL – a lot of extra work needs to be done to do this.

CPE dictionary replaces obsolete names with search patterns of CPE names. In fact, these are again CPE names according to [5]. Here in the ontology, this approach is rejected because there is no similar search engine in OWL. The ontology uses comprehensive referencing of CPE names (base names). Moreover, with this approach, the control over the integrity of the ontology is stronger using standard control tools such as HermiT reasoner that can be applied during the ontology loading in Protégé particularly.

The presented ontology is used in another real ontology developed that one for CVE vulnerabilities.

CPE ontology generator is published at https://github.com/VladimirDimitrov1957/CPE-ontology-generator.

8 Acknowledgements

I would also like to thank NIST for consulting on certain issues in the dictionary and especially to Amy Mahn for her responsiveness.

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

This research is supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science.

References

1. MITRE Corporation, CPE, https://cpe.mitre.org, last accessed 25/04/2021.
2. NIST, National Vulnerability Database, NVD, Official Common Platform Enumeration (CPE) Dictionary, https://nvd.nist.gov/products/cpe, last accessed 25/04/2021.
3. NIST, NISTIR 7695, Common Platform Enumeration: Naming Specification Version 2.3, https://csrc.nist.gov/publications/detail/nistir/7695/final, last accessed 25/04/2021.
4. NIST, NISTIR 7697, Common Platform Enumeration: Dictionary Specification Version 2.3, https://csrc.nist.gov/publications/detail/nistir/7697/final, last accessed 25/04/2021.
5. NIST, NISTIR 7696, Common Platform Enumeration: Name Matching Specification Version 2.3, https://csrc.nist.gov/publications/detail/nistir/7696/final, last accessed 25/04/2021.
6. NIST, NISTIR 7698, Common Platform Enumeration: Applicability Language Specification Version 2.3, https://csrc.nist.gov/publications/detail/nistir/7698/final, last accessed 25/04/2021.
7. NIST, National Vulnerability Database, NVD, https://nvd.nist.gov, last accessed 25/04/2021.
8. W3C, Naming and Addressing: URIs, URLs, ..., https://www.w3.org/Addressing, last accessed 25/04/2021.
9. ISO, ISO/IEC 14977: 1996(E), Information technology — Syntactic metalanguage — Extended BNF, https://www.iso.org/standard/26153.html, last accessed 25/04/2021.
10. IETF, RFC 5646, Tags for Identifying Languages, https://tools.ietf.org/html/rfc5646, last accessed 25/04/2021.
11. W3C, OWL 2 Web Ontology Language, Manchester Syntax (Second Edition, https://www.w3.org/TR/owl2-manchester-syntax, last accessed 25/04/2021.
12. Musen, M.A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003