Building annotated semantic model of software products towards integration of DBpedia with NVD vulnerability dataset

Andrei Brazhuk

***Abstract* - This work discusses integration of the DBpedia dataset with NVD (National Vulnerability Database) in order to bring some practical results to knowledge management in the field of software security.**

We have automatically mapped entities (software products and vendors), obtained from CPE (Common Platform Enumeration), with the corresponding elements of DBpedia, through the DBpedia Spotlight service. We have manually reviewed the annotation results and linked them into a semantic model. As NVD uses the CPE entities as a naming scheme for software products, the semantic model allows to identify NVD records, related to software products, mentioned in DBpedia; and can be used to extend DBpedia by vulnerabilities related data, and build advanced security models of software products. All the experimental models in the RDF format and Java-based software have freely been published by the GitHub service.

The mapping of NVD with DBpedia based on CPE and DBpedia Spotlight does not seem to be easy. The automatic annotation has suffered from getting general results, instead of specific ones. Also, there is an issue with possibility to choose the most general term in a given sequence. And the last challenge relates to possible incompleteness and inconsistency of the Linked Open Data. It needs to improve annotation techniques in order to involve fully automatic process there.

***Keywords* — software security, semantic annotation, knowledge management, DBpedia, DBpedia Spotlight, NVD, CPE, OWL API.**

I. Introduction

The idea of a Semantic Web is to extend the traditional Web of documents. In the new information space there are not only documents and links between them, but any entities (e.g. persons, organisations, countries, different kinds of products) and relations between them [1]. The Semantic Web is based on LOD (Linked Open Data). The LOD conception refers to best practices, guides, formats and technologies, that allow free publication and interlinking structured data on the Web. The LOD reality is formed from different public knowledge sources (datasets), available online through the Web-based services and as structured files. The most famous of them are Wikidata, DBpedia, YAGO, etc. (one can find the whole picture at https://lod-cloud.net).

We can consider DBpedia (https://wiki.dbpedia.org/), a crowd-sourced community project, as a most significant component of the LOD cloud and even as a key factor of the LOD conception success. The project has a powerful extraction framework and means, able to obtain the linked data from Wikipedia and its subprojects (mapping-based and raw infobox extraction, feature extraction, statistical extraction, NLP extraction). Also, it contains a lot of links to external entities, that gives it a role of hub for other LOD components [1].

The LOD cloud contains information about different aspects of human activity, in particular about software security, the last is considered as the main topic of this work. However, using the LOD data in the field of software security is not very common now. The most valuable sources of knowledge there can be treated as traditional sources, i.e. various directories of vulnerabilities, threats, and attacks, which aggregate different pieces of experience (positive and negative ones). It can be argued there are two groups of knowledge sources: primary sources, which contain information about security issues of end products; and secondary sources, which generalize the primary sources by the analysis and classification procedures in order to recognize typical challenges and form security guides and best practices. As primary sources can be considered CVE (Common Vulnerabilities and Exposures) or NVD (National Vulnerability Database), as dictionaries of publicly known vulnerabilities in software products; and CPE (Common Platform Enumeration) as a naming scheme for identification of software products, mentioned in NVD. CWE (Common Weakness Enumeration) and CAPEC (Common Attack Pattern Enumeration and Classification), as means for classification of cybersecurity weaknesses and attacks, might be considered as secondary sources.

The main advantage of the LOD approach to knowledge management is an opportunity to add pieces of intelligence to data processing. Although traditional data sources use well-structured data formats (XML, JSON), they do not have such capacity. Modern knowledge management systems are based on a semantic approach, that uses methods and technologies oriented to semantics, i.e. meaning of data. Usually a knowledge management system has an ontology (set of ontologies) as a core. Ontology-based systems use descriptive logics (a subset of first-order logics) as a background. The descriptive logic is able to depict concepts of a subject-specific area and relations between them in a very formal way. Reasoning procedures with relatively low computational complexity (under certain conditions) and

Manuscript received April 19, 2019.

Andrei Brazhuk is with the Yanka Kupala State University of Grodno, Grodno, Republic of Belarus (e-mail: brazhuk@grsu.by).

advanced rule-based processing can be added to that kind of systems. Existing LOD datasets, based on the RDF (Resource Description Framework) models can be combined with the OWL (Web Ontology Language) ontologies in order to implement advanced knowledge management systems.

This work discusses integration of DBpedia with NVD based on CPE in order to bring some practical results to knowledge management in the field of software security. We have automatically mapped (annotated) entities (software products and vendors), obtained from CPE, with the corresponding elements of DBpedia, through the DBpedia Spotlight service. We have manually reviewed the annotation results and linked them into a semantic model, aimed to connect the DBpedia and NVD datasets. The semantic model allows to obtain data from NVD, e.g. by SPARQL requests, that contain DBpedia entities. That might be useful to extend DBpedia by data, related to vulnerabilities, and to build advanced security models of software products (or groups of products).

Below a review of related researches is given, a structure of used semantic modes is depicted, a short description of an implementation is given, and main issues of automatic annotation named entities with DBpedia Spotlight are discussed.

Given models and software have freely been published by GitHub service (see link below).

II. Related work

Researches, related to creation new security semantic models (from scratch or based on traditional data sources), are quite perspective nowadays.

The work [2] has considered design of ontology-based data model as a part of network attack modelling for the SIEM (Security information and event management) systems. Proposed decisions are based on SCAP (Security Content Automation Protocol). In particular, they have described a common data model, which mentioned CAPEC and CWE; also, they have depicted an ontology of vulnerabilities, based on CVE.

A reference ontology for cybersecurity operational information proposed in [3]. They described operation domains, roles, databases and knowledge bases and relations between them and mapped existing traditional data sources to their ontology. Based on that work a mechanism of linking, locating, and discovering various cybersecurity information and its prototype have been proposed in [4]. They have claimed an ability of given means to discover and hold structured cybersecurity information over the Internet, i.e. to be a hub of web of cybersecurity.

The similar ideas have been brought by UCO (Unified Cybersecurity Ontology) [5], but with some improvements. The ontology has been based on the STIX (Structured Threat Information eXpression) specification and incorporates huge number of data and knowledge sources. The best result there might be that UCO is the first cybersecurity ontology, that has been mapped to the LOD cloud (DBpedia, Yago). Earlier [6] it was described the idea and framework for annotation cybersecurity terms and vulnerability descriptions extracted from NVD and different text sources with DBpedia Spotlight in order to create RDF-based security knowledge sources, linked to the LOD cloud. And the recent work [7] has described CCS (Cognitive Cybersecurity System) with incredible opportunities to ingest information from various textual sources and store them in a knowledge graph in order to derive improved actionable intelligence to different security application.

The work [8] has described an ontology based Software Security Tagger Framework, aimed to extract security concerns from textual information; that can yield several benefits (bug management, capturing zero day attacks, etc.).

Those ideas and means have inspired us strongly, in particular a challenge to unite the LOD cloud and traditional security data sources. This challenge can be considered as the very first step in a way to security data integration. That might rich the LOD cloud with NVD, CWE, CAPEC data; and it allows to use the LOD data by security systems, based on the traditional data sources.

However, most of the existing researches are mainly conceptual and declarative, their findings (e.g. used means, data sets) are described superficially. And the most disappointing thing is the practical results, that could be reused for new researches, are unavailable or only partially available (e.g. the UCO ontology has been published, however we have not managed to find a link to its RDF dataset).

So, that makes the idea of this work to bring some reusable results to knowledge management in the field of software security quite urgent.

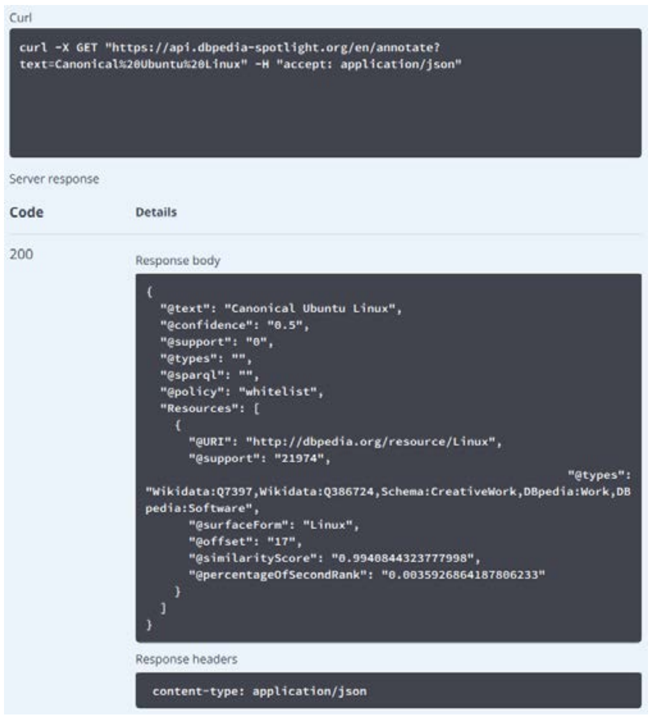
III. DBpedia and semantic annotation

Structured data of DBpedia available on the WEB as static datasets, through a public SPARQL endpoint (http://dbpedia.org/sparql) or in other ways (Faceted Web Service, REST API, third-part endpoints, etc.). Since the first public release in 2007 the DBpedia datasets had been updated once per year. The last static multi-language update (2016) included 13 billion RDF triples, and with the NIF (NLP Interchange Format) data they got the overall count of triplets to 23 billion. Also, live synchronization mechanism has been added to the DBpedia project, that processes updates in Wikipedia and the DBpedia ontology and allows third parties to update their copies of DBpedia [9].

The DBpedia dataset has own ontology (http://dbpedia.org/ontology or the “dbo” prefix). It is maintained by a community and intended to unify different data entities and provide effective data extraction from Wikipedia. Properties of entities are hold in the “dbp” namespace (http://dbpedia.org/property) and used primary by the raw infobox extraction. And the last namespace, called “dbr” (http://dbpedia/resource), is used to represent the Wikipedia pages. If a Wikipedia article exists, it means there is a DBpedia resource with a name, based on article’s title [9].

Semantic annotation is the task of identifying all relevant entities in a text document and linking them to a knowledge base [10]. This problem is usually divided to two ones: Semantic Annotation itself and Named Entity Disambiguation/Recognition (NED/NER). Semantic Annotation refers to a process of identification all entries from a knowledge base (i.e. named entities, abstract concepts, classes and properties) in a given text document; while NED/NER focuses on recognition and annotation named entities only.

There are a few freely available applications, aimed to annotate a text document with entities from DBpedia. For example, THD (Targeted Hypernym Discovery) [11] can recognize entities in text, written in English, German, Dutch languages, and enrich them with links from the Wikipedia, DBpedia and YAGO knowledge bases. Also, Marvin [12], a text annotator written in Java, can be used for annotation of text using multiple sources (WordNet, MetaMap, DBpedia, SKOS). Those projects (and others) might have some unique features, that would be useful for the semantic annotation task, but that requires additional research. As a dive into NLP has been out of scope of the work, we have got the conclusion, that the best option for us is to use DBpedia Spotlight [13,14]. The main criterion for us has been the opportunity to use it "out of box", so we have managed to perform the necessary research procedures through its public interface (https://api.dbpedia-spotlight.org) without deployment a local service.


Curl

```
curl -X GET "https://api.dbpedia-spotlight.org/en/annotate?
text=Canonical%20Ubuntu%20Linux" -H "accept: application/json"
```

Server response

Code Details

200

Response body

```
{
  "@text": "Canonical Ubuntu Linux",
  "@confidence": "0.5",
  "@support": "0",
  "@types": "",
  "@sparql": "",
  "@policy": "whitelist",
  "Resources": [
    {
      "@URI": "http://dbpedia.org/resource/Linux",
      "@support": "21974",
                                                          "@types":
"Wikidata:Q7397,Wikidata:Q386724,Schema:CreativeWork,DBpedia:Work,DB
pedia:Software",
      "@surfaceForm": "Linux",
      "@offset": "17",
      "@similarityScore": "0.9940844323777998",
      "@percentageOfSecondRank": "0.0035926864187806233"
    }
  ]
}
```

Response headers

```
content-type: application/json
```

Fig. 1. API request to Dbpedia Spotlight

DBpedia Spotlight is an open source project, developing a system for automatic annotation of text with the DBpedia entities. The annotation process consists of four stages. On the spotting stage they find phrases, that might indicate a DBpedia resource. Candidate selection allows to map the spotted phrases to several resources. The disambiguation stage is responsible for the best choice from the several candidates. Also, it is possible to customize results by adding filters. The full annotation procedure seems to be excess for the annotation of software products and vendors. However, DBpedia Spotlight does not have an option to distinguish semantic annotation and NER/NED [10]. They [10] are working to improve that by involving the FICLONE system, that combines named entity recognition system (Stanford NER) with the results of DBpedia Spotlight. However, that work is still in progress and there are not public available tools yet.

Figure 1 shows an example of a Spotlight API request and response. For responses the JSON format is used.

IV. CVE, NVD AND CPE

CVE (https://cve.mitre.org/) is a dictionary, that contains identification numbers, descriptions and external references for publicly known vulnerabilities in software products. Common identifiers make it easy to share the information across security databases and tools. CVE was launched by MITRE (https://cve.mitre.org) as a community effort in 1999. Since 2012 the CVE records have been stored in the CVRF (Common Vulnerability Reporting Framework) format, that is a very simple and laconic way to represent vulnerabilities as XML entities.

NVD (https://nvd.nist.gov) was launched by NIST (https://www.nist.gov/) in 2005. NVD extends CVE with the CVSS (Common Vulnerability Scoring System) metrics for numerical scores for vulnerabilities, the CWE (Common Weakness Enumeration) identifiers for their classification, and the CPE entities in order to identify vulnerable products. The NVD records are preferably saved in JSON (they are going to stop using XML). CVE and NVD are synchronized, total count of records is above one hundred thousand.

CPE (https://nvd.nist.gov/products/cpe) is a naming scheme and dictionary for software products. It is distributed in the XML format and regularly updated. CPE has some issues related to its flat structure. Its organization allows only to distinguish three classes of software (applications, hardware-specific, and operating systems). Also, there is no hierarchy of products, so adding a new version of a product causes creation of a new record with duplication of product and vendor data.

The last CPE 2.3 specification includes CPE naming scheme and CPE dictionary format. The CPE naming scheme is defined through well-formed names (WFN). Listing 1 shows the WFN template and record example. The part field represents a type of product (“a” - application, “h” - hardware, “o” - operating system). The asterisk symbol (*) means absence of a value.

Listing 1. WFN template and example

```
part:vendor:product:version:update:edition:lang:sw_ed:target_sw:t
arget_hw:other
o:canonical:ubuntu_linux:16.04:*:*:*:lts:*:*:*
```

One can get the current version of the CPE Dictionary as a compressed XML file from project webpage (https://nvd.nist.gov/products/cpe). Listing 2 shows an example of CPE record. WFN is an attribute of the “cpe-23:cpe23-item” tag, and full product name (often with version and update information) is in the “title” tag.

Listing 2. Example of CPE record

```
<cpe-item name="cpe:/o:canonical:ubuntu_linux:16.04.4">
<title xml:lang="en-US">Canonical Ubuntu Linux 16.04.4</title>
<references>
<reference href="http://releases.ubuntu.com/">
Version
```

```
</reference>
<reference href="http://old-releases.ubuntu.com/releases/">
Version
</reference>
</references>
<cpe-23:cpe23-item
name="cpe:2.3:o:canonical:ubuntu_linux:16.04.4:*:*:*:*:*:*:*"/>
</cpe-item>
```

The NVD database is represented as a set of compressed JSON files (each one contains vulnerabilities for particular year, starting from 2002, and there is a file, that contains the last changes). One can get the current version of NVD from its data feeds webpage (https://nvd.nist.gov/vuln/data-feeds).

Vendor and product information might be included to a CVE entity as a part of the “affects” field (Listing 3) or the "configurations" field (Listing 4). They use the logical and comparison operators to strictly define versions of affected products.

Listing 3. Part of “affects” field of CPE record

```
{
  "vendor_name" : "canonical",
    "product" : {
    "product_data" : [ {
    "product_name" : "ubuntu_linux",
    "version" : {
    "version_data" : [ {
    "version_value" : "16.04",
    "version_affected" : "="
    }, {
    "version_value" : "17.10",
    "version_affected" : "="
    } ]
    }
  } ]
}
```

Listing 4. Part of “configurations” field of CPE record

```
{
"operator" : "OR",
"cpe_match" : [ {
"vulnerable" : true,
"cpe23Uri" :
"cpe:2.3:o:canonical:ubuntu_linux:16.04:*:*:*:lts:*:*:*"
}, {
"vulnerable" : true,
"cpe23Uri" :
"cpe:2.3:o:canonical:ubuntu_linux:17.10:*:*:*:*:*:*:*"
} ]
}
```

V. Structure of semantic model

One of the purposes of this work has been to annotate vendor and software entities from the CPE Dictionary. Informally, to do that, we had to apply the following procedure to the CPE data. To get a DBpedia entity for a software product it is necessary to take a string from the “title” tag of a CPE record, e.g. “Canonical Ubuntu Linux 16.04.4” (Listing 2), cut version and update pieces of data in order to get a clear name, e.g. “Canonical Ubuntu Linux”, and try to annotate the clear name. Similarly, to get a DBpedia entity for a vendor, e.g. “canonical”, it is necessary to take a vendor field from the “cpe-23:cpe23-item” string (Listing 2) and try to annotate it.

In order to store given results a semantic model has been created, shown in Figure 2.

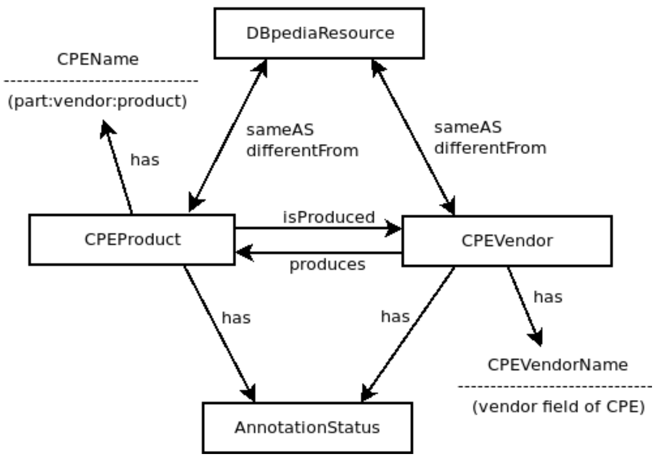


Fig. 2. Semantic model of CPE/DBpedia software vendors and products

“DBpediaResource” entities show mappings between different DBpedia objects and entities, obtained from CPE (“CPEVendor” and “CPEProduct”). DBpediaResource can be the “differentFrom” or “sameAs” a CPEVendor or CPEProduct. The property of difference might be added by hand in order to correct possible annotation errors. The “sameAs” properties mainly have to be obtain from the automatic annotation process, but some of them might be manually added too.

Obviously, parsing of the CPE Dictionary allows to determine relations between vendors and products, like “CPEVendor produces some CPEProduct”, and “CPEProduct is produced by some CPEVendor”. These properties (“produces” and “isProduced”) are actually asymmetric, so there is an option to get one from another by a reasoning process.

“AnnotationStatus” entities are intended to describe annotation states of the CPEProducts and CPEVendors. It might have successfully annotated CPE entity, false annotated, manually annotated, or with “annotation not found” state.

Also, CPE style names are used to identify products and vendors, mentioned in the NVD dataset. For CPEProduct it is “CPEName” in format “part:vendor:product”. For CPEVendor the “vendor” field from CPE string is used. An alternative way to solve the identification issue is to give the same identifiers (IRI) for entities from the CPE and NVD datasets.

The development of fully-functional semantic model of NVD has been out of scope of this work. To illustrate given ideas, we have created the simple NVD model (with an intention to expand the model in future research), shown in Figure 3.

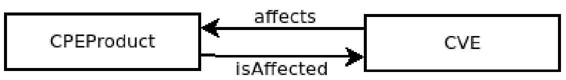


Fig. 3. NVD semantic model

A “CVE” entity represents a CVE vulnerability, and it affects CPEProduct, that represents an entity from the CPE/DBpedia model; so the CPEProduct is affected by the CVE (might be obtained by reasoning).

VI. Implementation of semantic model

The proposed implementation includes two software modules: for creation of the CPE/DBpedia semantic model and for creation of the NVD model. Also we consider the CPE/DBpedia and NVD models as end products, especially the first one, because manual checking and corrections have been performed for automatically annotated entities to increase quality of the results. The source code of the developed software modules and the RDF files of resulting models have freely been published by Github (https://github.com/nets4geeks/abCAPECCWESemanticModel).

Software modules have been written in Java. The first one parses the CPE Dictionary in the XML format, annotates the CPE entities (products and vendors) by asking DBpedia Spotlight, writes the results to a RDF file (CPEDBpedialModel.ttl). The second one parses the NVD files in the JSON format and writes results to a RDF file (NVDSemanticModel.ttl). To read XML the standard package javax.xml.parsers has been used; it contains API, able to manipulate an object model (DOM - Document Object Model) of XML. To read JSON the Jackson JSON parser (https://github.com/FasterXML/jackson-core) has been used, it can combine line-by-line parsing with using object model. The OWL API version 5 external library (https://github.com/owlcs/owlapi) has been used to generate RDF. The Apache Maven (https://maven.apache.org/) is responsible for deployment and building of the software.

CPEDBpedialModel.ttl contains the CPE/DBpedia semantic model, i.e. facts about annotated DBpedia entities (Listing 5), CPE products (Listing 6), and CPE vendors (Listing 7).

Listing 5. Example of DBpediaResource (CPEDBpedialModel.ttl)
```
<http://dbpedia.org/resource/Ubuntu_(operating_system)>
   rdf:type owl:NamedIndividual ,
  :DBpediaResource ;
  owl:sameAs :canonical_ubuntu .
```

Listing 6. Example of CPEProduct (CPEDBpedialModel.ttl)
```
:canonical_ubuntu_linux rdf:type owl:NamedIndividual ,
   :CPEProduct ;
   :hasAnnotationStatus :ManualAnnotatedStatus ;
   :isProducedByVendor :canonical_vendor ;
   :hasCPEName "o:canonical:ubuntu_linux"@en ;
   rdfs:comment "Canonical Ubuntu Linux "@en ;
   rdfs:label "canonical:ubuntu_linux"@en .
```

Listing 7. Example of CPEVendor (CPEDBpedialModel.ttl)
```
:canonical_vendor rdf:type owl:NamedIndividual ,
   :CPEVendor ;
   :hasAnnotationStatus :ManualAnnotatedStatus ;
   :producesProduct :canonical_accountsservice ,
   :canonical_acpidashsupport ,
   :canonical_juju ,
   :canonical_libpamdashmodules ,
   :canonical_ltsp_display_manager ,
   :canonical_lxcfs ,
   :canonical_php5 ,
   :canonical_reportbug ,
   :canonical_softwaredashproperties ,
   :canonical_telepathydashidle ,
   :canonical_ubuntu_core ,
   :canonical_ubuntu_linux ,
   :canonical_ubuntu_touch ,
   :canonical_updatedashmanager ;
   :hasCPEVendorName "canonical"@en ;
   rdfs:label "canonical"@en
```

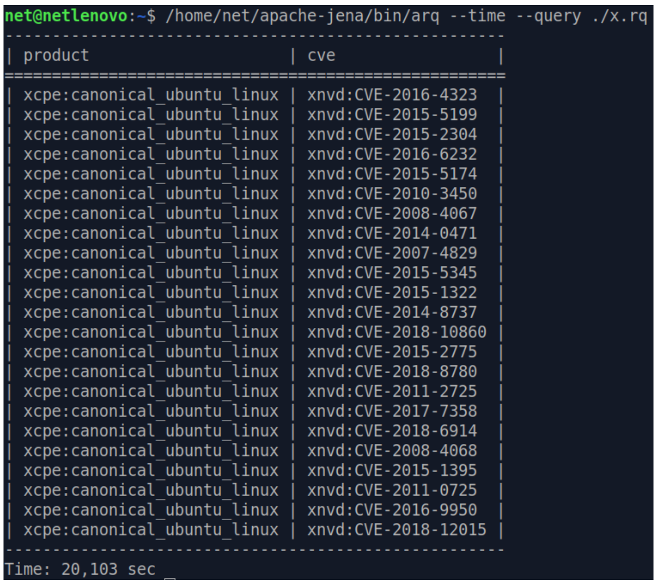
NVDSemanticModel.ttl contains the simplest NVD semantic model, that allows to unite CPE product entities with CVE entities. Listing 8 shows an example of a CVE entity.

Listing 8. Example of CVE (NVDSemanticModel.ttl)
```
:CVE-2017-7358
   rdf:type owl:NamedIndividual ,
   :CVE ;
   :affectsProduct
<http://www.grsu.by/net/CPEDBpediaModel#canonical_ubuntu_linux> ,
<http://www.grsu.by/net/CPEDBpediaModel#lightdm_project_lightdm> ;
   :problemsCWE
<http://www.grsu.by/net/CAPECCWEAttackPatterns#iCWE_22> ;
    rdfs:label "CVE-2017-7358"@en .
```

If one has got CPEDBpedialModel.ttl and NVDSemanticModel.ttl, it can test the models using SPARQL with the Apache Jena tool set (https://jena.apache.org/). For example to get all the CVE vulnerabilities of the DBpedia resource with the identifier http://dbpedia.org/resource/Ubuntu_(operating_system) (or *dbr:Ubuntu_(opearating_system)*), that relate to the “CWE 22” weakness, one can use a SPARQL request, shown in Listing 9 (the “problemsCWE” property has been added to the NVD model through additional research, also see details about the CWE/CAPEC model in [15]).

Listing 9. Example of using CPE/DBpedia semantic model
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
PREFIX owl: <http://www.w3.org/2002/07/owl#>.
PREFIX xcpe: <http://www.grsu.by/net/CPEDBpediaModel#>.
PREFIX xnvd: <http://www.grsu.by/net/NVDSemanticModel#>.
PREFIX xcwe: <http://www.grsu.by/net/CAPECCWEAttackPatterns#>.
PREFIX dbr: <http://dbpedia.org/resource/>.
SELECT ?product ?cve
FROM <./CPEDBpediaModel.ttl>
FROM <./NVDSemanticModel.ttl>
WHERE
{
  dbr:Ubuntu_\(operating_system\) owl:sameAs ?product .
  ?cve xnvd:affectsProduct ?product ;
   xnvd:problemsCWE xcwe:iCWE_22 .
}
```

The execution of the SPARQL request from Listing 9, is shown in Figure 4.



```
net@netlenovo:~$ /home/net/apache-jena/bin/arq --time --query ./x.rq
-----------------------------------------------------
| product                      | cve                |
=====================================================
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2016-4323  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-5199  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-2304  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2016-6232  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-5174  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2010-3450  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2008-4067  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2014-0471  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2007-4829  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-5345  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-1322  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2014-8737  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2018-10860 |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-2775  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2018-8780  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2011-2725  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2017-7358  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2018-6914  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2008-4068  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2015-1395  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2011-0725  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2016-9950  |
| xcpe:canonical_ubuntu_linux | xnvd:CVE-2018-12015 |
-----------------------------------------------------
Time: 20,103 sec _
```

Fig. 4. Execution of SPARQL request

VII. Discussion of results

Creation of the CPE/DBpedia semantic model was actually passing with two stages: after the automatic annotation had been done, we performed the manual review of given results in order to analyse them and make some improvements. Obviously, the automatic annotation process might produce some mistaken results. A problem with DBpedia Spotlight is they do not recommend it for short pieces of text, so we used this service at own risk.

We had also expected some generalized results. For example, a most general result for "Ubuntu" is a Southern African philosophy or *dbr:Ubuntu_(philosophy).* To reduce influence of generalization mistakes we built into the software a set of restriction rules for the automatic annotation results, shown in Table 1.

Table I
Restrictions for automatic annotation

Type of entity	Include rdf:type	Exclude rdf:type
Product "a" (application)	dbr:Software	-
Product "o" (operating system)	dbr:Software	-
Product "h" (hardware)	dbr:Device	dbr:Weapon dbr:Engine dbr:Instrument
Vendor	dbr:Company dbr:Organisation dbr:Developer	-

A product "a" means a product, having the "a" part of its CPE name (or an application), "o" means an operating system, "h" points to a hardware specific product. The rules are based on the rdf:type properties of a DBpedia entity. For particular type of entity include rules act before exclude rules. So, a hardware entity is only taken into consideration, if it is a device and not weapon, engine or instrument.

Below some distinctive examples of annotation errors are shown, that have been found during the manual review.

Applying the restriction rules to the automatic annotation process has avoided many generalization mistakes, e.g. refusing the recognition of the Juniper Router M10 as the M10 tank destroyer of World War II (*dbr:M10_tank_destroyer*). However, taking into consideration the not strict structure of DBpedia entities, some valuable annotations might have been lost, because of absence of necessary properties (e.g. "rdf:type dbr:Software" for an application or operating system).

The next kind of annotation mistakes, we have managed to recognize, refers as a choice of the most general term in a sequence. An example of such failure is shown in Figure 1. Spotlight had recognized the "Canonical Ubuntu Linux" term ("Ubuntu Linux" too) as more general Linux (*dbr:Linux*); so, it required a manual correction (see Listings 5-6). Also many of Linux-related entities (Adobe Flash player for Linux, Gentoo Linux, Gnome, GNU libc, Linux kernel, SUSE Linux, etc.) have been mapped to the most general Linux entity. And the similar issue is with Drupal (and WordPress too). There are plenty CPE products, having the names "something for Drupal" (third-part addons, plugins, etc.), that have been recognized as Drupal (*dbr:Drupal*).

For hardware a most distinctive example of false annotation is mapping of the IBM/Lenovo Flex system nodes (e.g. "IBM Flex System X220 M4 Firmware") as Apache Flex (*dbr:Apache_Flex*). The reason of failure is probably Wikipedia does not have a page, related to IBM/Lenovo Flex. More common problem here is possible ambiguous interpretation of hardware-related entities, it might be a confusion how to consider, saying, a piece of firmware: as hardware or software (operating system) entity.

The most inexplicable failure with vendors is about inability to annotate "microsoft" (the leading lowercase letter). However, Spotlight can map "Microsoft" (the leading capital letter) to the right entity (*dbr:Microsoft*). The service does not seem to be case-sensitive, because it is able to annotate the "cisco" (the leading lowercase) vendor in the right way (*dbr:Cisco_Systems*).

And again, Spotlight had been unable to annotate the "canonical" vendor (as well as "Canonical"), and the mapping was added manually.

Table 2
Annotation summary of semantic model

Type of entity	Found entities	Automatic annotated	False annotated	Manual annotated
Product "a" (application)	13143	3623	206	0
Product "o" (operating system)	1394	198	49	9
Product "h" (hardware)	4515	119	44	0
Vendor	5596	387	28	4

Table 2 shows the summary of annotations, which the current version of the CPE/DBpedia semantic model contains. “Found entities” represents the number of entities, obtained from CPE; “Automatic annotated” shows the number of automatic annotated entities by Spotlight; “False annotated” and “Manual annotated’ describes the results of the manual review. Keep in mind, the count of manual annotations might change, because we are updating the model in order to improve it (CPEDBpediaModelManual.ttl).

Taking into consideration the relatively low success rate of annotations, the most significant result, that follows from Table 2, is the LOD cloud contains a little information about software products from the CPE Dictionary at the moment. That is a quite rough estimate, but, obviously, effective integration of the LOD and NVD data based on CPE is only possible for a limited set of software products. Also, we believe that if someone replaces our rough automatic annotation approach with a more accurate one, it might be possible to get slightly better results.

VIII. Conclusions

This work shows the process of the creation, description of the structure of the semantic model of software products and vendors, that maps DBpedia and NVD datasets by the CPE data and DBpedia Spotlight annotation results. We believe it contains the qualitative data, because it has been checked manually. The semantic model can be used to extend the DBpedia dataset by vulnerabilities related data from the NVD software security repository. Given model and software has freely been published with the public GitHub service (see link above).

The mapping of NVD with DBpedia based on CPE and DBpedia Spotlight as the very first step to security data integration does not seem to be easy. The automatic annotation has strongly suffered from getting general results, instead of specific ones. Also, there is an issue with possibility to choose the most general term in a given sequence. And the last challenge relates to possible incompleteness and inconsistency of the LOD data. Obviously, it needs to improve annotation techniques in order to involve fully automatic process there.

Also it should be taken into consideration, DBpedia resources should be treated as dynamic entities with changeable data and structure. A thing, we call a DBpedia resource and use to create an object of a knowledge base, is actually a Wikipedia page, managed by different people. It is not protected from violations of structure and mistaken data, and it can be assumed possibility of data corruption with malicious goal. So, a security system, that uses the LOD data, should be preserved from that kind of intrusion.

References

[1] Färber M. et al. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago //Semantic Web. – 2018. – T. 9. – №. 1. – C. 77-129.
[2] A. A. Chechulin, I. V. Kotenko, O. V. Polubelova, Design of the ontology based data model for the network attack modeling system., Trudy SPIIRAN., 2013., T. 26., pp. 26-39.
[3] Takahashi T., Kadobayashi Y. Reference ontology for cybersecurity operational information //The Computer Journal. – 2015. – T. 58. – №. 10. – C. 2297-2312.
[4] Takahashi T. et al. Web of cybersecurity: Linking, locating, and discovering structured cybersecurity information //International Journal of Communication Systems. – 2018. – T. 31. – №. 3. – C. E3470.
[5] Syed Z. et al. UCO: A Unified Cybersecurity Ontology //AAAI Workshop: Artificial Intelligence for Cyber Security. – 2016.
[6] A. Joshi, R. Lal, T. Finin, and A. Joshi, “Extracting Cybersecurity Related Linked Data from Text” in IEEE Seventh International Conference on Semantic Computing, 2013, pp. 252–259.
[7] Narayanan S. N. et al. Early Detection of Cybersecurity Threats Using Collaborative Cognition //2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC). – IEEE, 2018. – C. 354-363
[8] Alqahtani S. S., Rilling J. An ontology-based approach to automate tagging of software artifacts //Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. – IEEE Press, 2017. – C. 169-174.
[9] Lehmann J. et al. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. //Semantic Web. – 2015. – T. 6. – №. 2. – C. 167-195.
[10] Chabchoub M., Gagnon M., Zouaq A. FICLONE: improving DBpedia spotlight using named entity recognition and collective disambiguation //Open Journal of Semantic Web (OJSW). – 2018. – T. 5. – №. 1. – C. 12-28.
[11] Kliegr T. Linked hypernyms: Enriching DBpedia with Targeted Hypernym Discovery, Journal of Web Semantics, JWS, Elsevier, 2015.
[12] Milosevic N. Marvin: Semantic annotation using multiple knowledge sources // arXiv preprint arXiv:1602.00515. – 2016.
[13] Mendes P. N. et al. DBpedia spotlight: shedding light on the web of documents. //Proceedings of the 7th international conference on semantic systems. – ACM, 2011. – C. 1-8
[14] Daiber J. et al. Improving efficiency and accuracy in multilingual entity extraction. //Proceedings of the 9th International Conference on Semantic Systems. – ACM, 2013. – C. 121-124.
[15] Brazhuk A. Semantic model of attacks and vulnerabilities based on CAPEC and CWE dictionaries. //International Journal of Open Information Technologies. – 2019. – T. 7. – №. 3. – C. 38-41

Andrei Iosifovich BRAZHUK
Researcher, senior lecturer at the Yanka Kupala State University of Grodno (https://www.grsu.by)
email: brazhuk@grsu.by
publications: https://scholar.google.com/citations?user=lxR8RLkAAAAJ