Framework for ontology-driven threat modelling of modern computer systems

Andrei Brazhuk, Evgeny Olizarovich

Abstract - **Threat modelling of a computer system is based on the system analysis of its architecture on early development stages (requirements, design) and creation of a threat model that represents security aspects of the system (threats and mitigations). Used in this field means, like data flow diagrams (DFD) and Application threat modelling approach (OWASP, Microsoft), are mainly informal and hard to involve automation.**

In order to overcome these restrictions, we have created the ontology-driven threat modelling (OdTM) framework, which allows to use graphical notation of DFD diagrams and semantic domain-specific threat models to build threat models for different computer systems. Each domain-specific threat model has a set of typical components of some subject area and threats/countermeasures associated with these components. An end user can describe a computer system with DFD diagram(s), then reasoning procedures are able to build a threat model for that system.

The OdTM framework consists of a common approach of the architectural security analysis and method of semantic interpretation of DFD diagrams and automatic reasoning of relevant threats and countermeasures. We have developed the base threat model as OWL (Web ontology language) ontology that enables creation of domain-specific threat models as OWL ontologies and extension them with different external knowledge sources (knowledge "mining", the Linked Open Data etc.). To illustrate proposed approach, we have used a semantic version of a model that depicts common threats against cloud computer systems.

Keywords — **software security, knowledge management, threat modelling, OWL, DFD.**

I. Introduction

Threat modelling is a process of identification of security threats and their countermeasures, aimed to increase security level of computer systems. Common approach of the threat modelling is based on the analysis of structure and organization of a computer system (i.e. its architecture) on the early stages of its development (requirements, design) and building a threat model that represents all the security aspects of the system.

There are some challenges with the threat modelling. Its practices and methods are principally informal and involving of formalization and automation is quite hard there [1]. Current means in this field give a development team flexible features to define, consider, document, evaluate, and discuss threats. However, there is a lack of formal approaches to describe a computer system architecture and structured knowledge sources about threats and their mitigations, which can be used for the threat modelling automation.

To model computer infrastructure, it has been proposed the Infrastructure and Network Description Language (INDL) and Network Markup Language (NML) [2], which provide ontology-based technology independent descriptions of computer systems (processing, storages, network topologies, virtualization etc.); however, their approach has not become common and there is no tool that supports it now. Another option is the Common Information Model (CIM), which is an open standard (https://www.dmtf.org/standards/cim), defining how managed elements in IT environment and relationships between them are represented as common objects. The CIM means are used on the operation stage of system life cycle (e.g. Windows Management Instrumentation - WMI), and its adoption to the design stage seems to be a challenge. So, the most common approach, used for the security modelling purposes, is Data Flow Diagrams (DFD) [1], which can be considered as a way of informal graphical representation of system architecture.

To describe threats, the special catalogues and languages are used like Common Attack Pattern Enumeration and Classification (CAPEC), Common Weakness Enumeration (CWE), Common Vulnerabilities and Exposures (CVE), WASC Threat Classification, OWASP, Structured Threat Information eXpression (STIX). These means are intended to depict threat structure and taxonomy and provide answering the questions like what threats may be associated with a particular design decision, which design decisions are more suitable to mitigate current threats etc. Most of the mentioned above knowledge sources are quite informal and required revision of their terminology; for example, none of them operates with the "threat" term. So, it can be a challenge to correlate threats, expressed with end user terminology, into attacks, weaknesses and vulnerabilities, described by the existing threat catalogues and CTI (Cyber Threat Intelligent) systems.

In order to bring a formal approach to this filed we have created the ontology-driven threat modelling (OdTM) framework, which includes a common approach of the architectural security analysis, method of semantic interpretation of DFD diagrams and automatic reasoning of relevant threats and countermeasures; that includes the base threat model, which enables creation of domain-specific threat models. Each domain-specific model has a set of typical components of some subject area (e.g. Cloud-based computer systems) and threats/countermeasures associated

Manuscript received October 30, 2019.

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

with these components. An end user can describe its computer system in terms of a domain-specific model with DFD diagram(s), then the reasoning procedures are able to build a threat model for the system. Mathematical background of used means is based on the descriptive logics (DL) as a subset of the first-order logics, which have an ability to describe concepts of domain-specific area and relations between them in very formal way; also that enables automatic reasoning features with relatively low computational complexity. For practical tasks OWL (Web Ontology Language) can be used, which has initially been created for the Semantic WEB and can be used for any knowledge-based system.

To illustrate proposed approach, we have used a semantic version of the Common Cloud Computing Threat Model (CCCTM), which is intended to depict common threats against cloud computer systems from the architectural point of view.

II. Related work

The Application threat modelling approach has been introduced by the OWASP (Open Web Application Security Project) organization. Their vision of the threat modelling process includes three stages: Decompose the Application, Determine and rank threats, Determine countermeasures and mitigation. On the first stage a development team can use a simple representation of application structure based on DFD diagram(s). This variation of DFD consists of three types of objects, like “External Entity”, “Process”, “Data Store”; the “Data Flow” type is used to describe data exchange between objects; and the “Trust Boundary” type allows to show borders between groups of objects (Fig. 1). Using the DFD diagram(s) on the next stages the development team can build an informal threat model of the application through discussion and usage the information of existent threat catalogues.

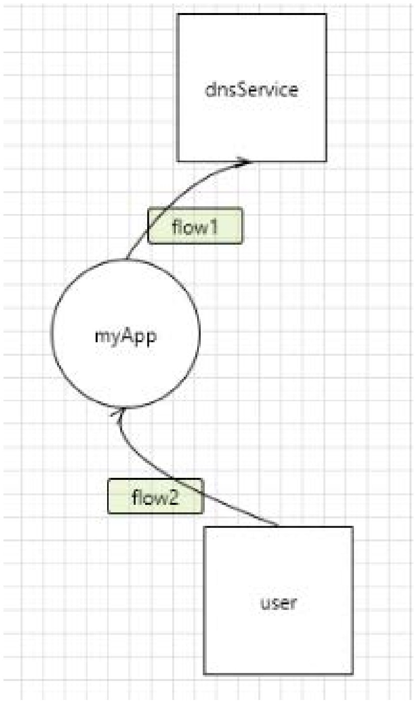
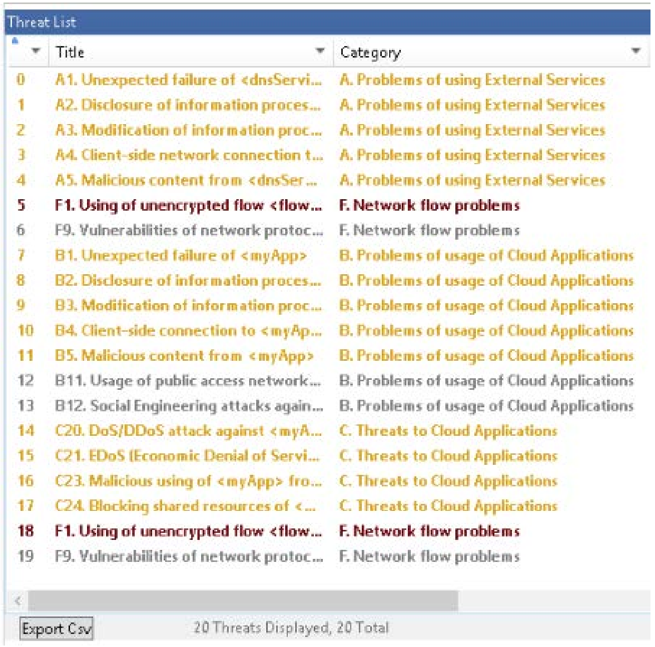


Fig. 1. A simple DFD diagram

It is known two ways of automation of this field. The first one deals with automatic building of DFD diagrams from source code. The tools like PyTM (pypi.org/project/pytm) and Threatspec (threatspec.org) utilize the following idea: during coding process a programmer adds special notes to source codes, after that a special tool should be run that analysis the notes and creates a DFD diagram.

The second way of the automation deals with analysis of DFD diagrams. Microsoft has added a little automation there with the Threat Modelling (TM) tool. The Microsoft TM software consists of a drag-and-drop DFD editor, simple rule-based reasoner, report subsystem, and built-in threat template editor. Threat template describes a threat model of particular domain-specific area with relevant enumerations of DFD items and threats related to these items. Using the model template an end user is able to depict its system as a DFD diagram (Fig. 1), and then automatically obtain a list of the relevant threats (Fig. 2).


Threat List

	Title	Category
0	A1. Unexpected failure of <dnsServi...	A. Problems of using External Services
1	A2. Disclosure of information proces...	A. Problems of using External Services
2	A3. Modification of information proc...	A. Problems of using External Services
3	A4. Client-side network connection t...	A. Problems of using External Services
4	A5. Malicious content from <dnsSer...	A. Problems of using External Services
5	**F1. Using of unencrypted flow <flow...**	**F. Network flow problems**
6	F9. Vulnerabilities of network protoc...	F. Network flow problems
7	B1. Unexpected failure of <myApp>	B. Problems of usage of Cloud Applications
8	B2. Disclosure of information proces...	B. Problems of usage of Cloud Applications
9	B3. Modification of information proc...	B. Problems of usage of Cloud Applications
10	B4. Client-side connection to <myAp...	B. Problems of usage of Cloud Applications
11	B5. Malicious content from <myApp>	B. Problems of usage of Cloud Applications
12	B11. Usage of public access network...	B. Problems of usage of Cloud Applications
13	B12. Social Engineering attacks again...	B. Problems of usage of Cloud Applications
14	C20. DoS/DDoS attack against <myA...	C. Threats to Cloud Applications
15	C21. EDoS (Economic Denial of Servi...	C. Threats to Cloud Applications
16	C23. Malicious using of <myApp> fro...	C. Threats to Cloud Applications
17	C24. Blocking shared resources of <...	C. Threats to Cloud Applications
18	**F1. Using of unencrypted flow <flow...**	**F. Network flow problems**
19	F9. Vulnerabilities of network protoc...	F. Network flow problems

20 Threats Displayed, 20 Total

Fig. 2. List of threats in Microsoft TM

Threat template is XML document that contains list of possible components (stencils) of system architecture and threat descriptions (threat types). Stencils can form two-level hierarchy with derived stencils; and threat types can be classified into categories. Each stencil has a set of properties, which can have fixed values (constraints).

Each threat type has a set of properties too; and flow-based rules can be associated with a threat type both for including it in a list and ignoring the threat for particular application. Threat rule can be “attached” to flow object. Microsoft uses a simple rule language to describe these rules. Each flow has starting object (source) and ending object (target). Using these objects, it might be possible to create a rule like:

target is [External Service] and flow is [Data Flow]

where “External Service” and “Data Flow” are examples of stencils.

Also properties of objects can be used in rules, like:

target.[service_responsibility] is 'uncontrolled' and flow.[flow_type] is 'data'

where "service_responsibility" and "flow_type" are properties of the stencils. The language also allows to say that some flow crosses some boundary; and you can use logical primitives ("and", "or", "not").

There have been some works, dedicated to research and creation of domain-specific models based on DFD, like Software define networks (SDN) [3], health systems [4, 5], and real-time web-conferencing [6].

Using the Microsoft TM tool and threat rule language we also used to implement the Common Cloud Computing Threat Model (CCCTM). It describes basic threats against cloud computer systems from the architectural point of view. Proposed threat template has freely been published with the GitHub service (github.com/nets4geeks/CCCTM_template) and Fig. 1-2 show an example of the CCCTM usage. The CCCTM research has found out couple issues with the Microsoft implementation. Their tool operates with two level hierarchy of stencils, however for description of complex computer systems it usually requires more layers of abstraction. Also there is a lack of full-featured mitigation hierarchy, which would allow a user to choose a mitigation to a threat from a relevant list.

There are few researches aimed to improve the Microsoft approach. For example, the work [7] has proposed an approach to architectural risk analysis that leverages the threat modelling by introduction of extended Data Flow Diagrams (EDFD), declaring a few improvements to DFD, e.g. decomposition of diagram components, bidirectional channels; provision of a threat pattern catalogue in a machine readable form (their knowledge base uses a domain-specific rule language, based on a graph query language); and creation of a visual EDFD viewer. That work contains great results, however in our opinion the OWL and DL background, as an implementation of object-oriented design approach to the threat modelling, has a few advantages (e.g. better representation for users, stricter formalization) over the structured methods used by the graph approach.

III. Ontology-driven threat modelling approach

The Ontology-driven threat modelling (OdTM) approach is intended to improve the architectural security analysis of computer systems by involving formalization and automation of its procedures. On the top would be a system, that allows an end user to depict different aspects of its application with DFD diagrams and automatically identify security threats and process them (including finding the relevant mitigations). Behind that there are ontology-driven models and methods that enable semi-automatic building of domain-specific threat models, creation of libraries of DFD components and threat related to them, and analysis of user DFD diagrams.

Fig. 3 shows the structure of the OdTM approach. *Base threat model*, implemented as OWL ontology, enables necessary automatic reasoning features and contains basic concepts and individuals, representing components of DFD diagrams, threats, mitigations, and their properties.

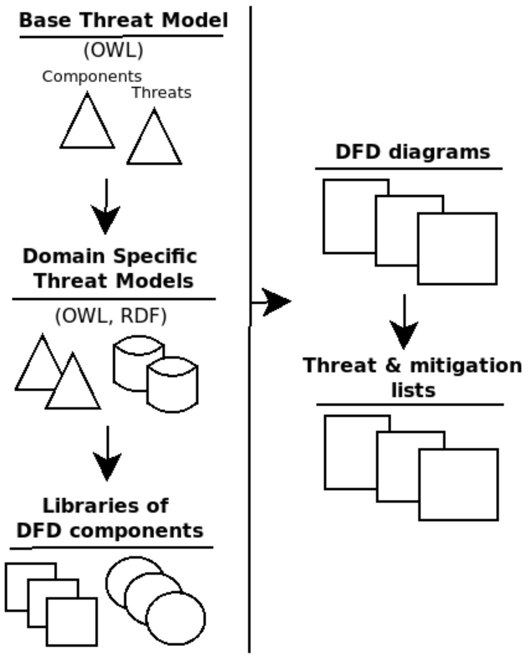

Base Threat Model
(OWL)
Components
Threats
Domain Specific Threat Models
(OWL, RDF)
Libraries of DFD components
DFD diagrams
Threat & mitigation lists

Fig. 3. Main components of the OdTM framework

If extend the basic model with specific components, threats and mitigations, related to an architecture of particular type of computer system, it can be possible to create a *domain-specific threat model.* There are number ways to enrich the domain-specific security knowledge: manually by experts, with mining knowledge from unstructured and structured traditional sources (e.g. attack, threat, vulnerability catalogues), and usage of semantic structured sources, in particular, the LOD (Linked Open Data) cloud. Automatic data mining and the LOD integration allow to keep the relevance of the domain-specific threat models. Domain-specific threat models are represented as OWL ontologies too, but they can be connected with different external linked data sources (other OWL ontologies and RDF data sets).

An end user depicts its computer system as *DFD diagram(s).* Then a set of reasoning and inferring procedures are able to automatically build *lists of relevant threats and mitigations* from those graphical user interpretations of the target computer system and domain specific threat model.

IV. Semantic interpretation of DFD diagrams

The OdTM base threat model as OWL ontology enables semantic interpretation of DFD diagrams and automatic building of threat/countermeasure lists by reasoning features. The ontology in the functional syntax has been freely published as the *OdTMBaseThreatModel.owl* file with the GitHub service (github.com/nets4geeks/OdTM).

According common approach (OWASP, Microsoft) DFD diagram, used in architectural security analysis, should consist of "Stencils" (Fig. 4): "Targets" represent architectural components, connected by directional flows ("DataFlows"), which might be restricted by "Trust Boundaries". There are three types of Targets in the classical model: "Processes", external entities ("ExternalInteractor") and "DataStores". Boundaries can be like lines ("TrustLineBoundary") and areas ("TrustBorderBoundary").

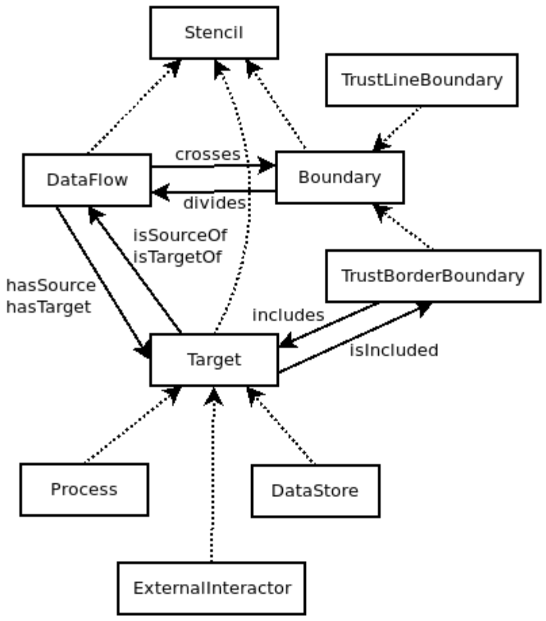


Fig. 4. Stencils and their properties of OdTM base threat model

Data Flow concept is defined by its starting and ending edges (Source and Target), line boundaries crossed by it, and “border” boundaries, in which it is included. To model the edges, the “hasSource”and “hasTarget” properties are used (it is supposed that their ranges are Targets), to model line boundaries the “crosses” property is used (range is “TrustLineBoundary”). The “includes” property has the “TrustBorderBoundary” as a domain and Target as a range (it requires a root box, to which all the items should be included by default).

To represent DFD diagram we should populate the ontology with instances. The description of a data flow instance and its edges is like:

Process (pr1) (1)
Process (pr2)
DataFlow (flow)
hasSource (flow, pr1)
hasTarget (flow, pr2)

where “flow” is an instance of the DataFlow concept, and “pr1”, “pr2” are instances of the “Process” concept. The “hasSource (flow, pr1)” phrase tells “flow” has source “pr1”.

The applying of a line boundary is like:

TrustLineBoundary (line) (2)
crosses (flow, line)

The applying of a border boundary is like:

TrustBorderBoundary (box) (3)
includes (box, flow)

Ontology-based reasoning allows to get some extra information from the example of data flow. So using inverse properties “isSourceOf” and “isTargetOf” to “hasSource” and “hasTarget”, it is determined that “pr1” is a source of “flow”, and “pr2” is its target. Also through the “divides” property (inverse to “crosses”) it argued that “line” divides “flow”; and the “isIncluded” property is inverse to “includes” and tells that “flow” is included into “box”.

V. Automatic reasoning threats and countermeasures

It can be argued that components of a computer system mainly suffer from remote threats, so threats should be applied to data flows. The “Threat” concept (Fig. 5) from the base model “affects” some data flow (the domain is supposed to be “Threat”, and range as “DataFlow”). Also the inverse property called “isAffectedBy” is used.

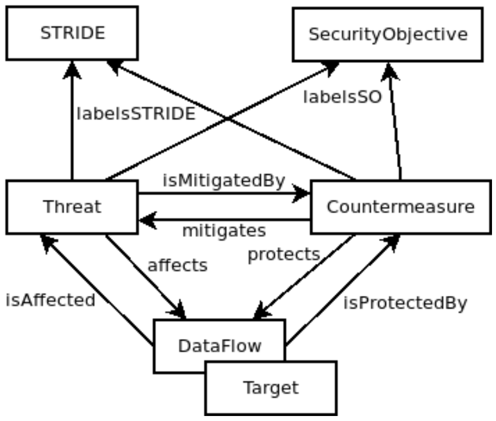


Fig. 5. Threats and countermeasures of OdTM base threat model

A flow template for a particular threat should be defined by the “hasSource” and “hasTarget” properties like:

$$Template1 \equiv DataFlow \cap \exists hasSource.Process \cap \exists hasTarget.Process \cap \exists crosses.TrustLineBoundary \quad (4)$$

(the “≡” symbol tells a concept is a precisely described here thing; “∩” is a conjunction; and “∃” means has a property with appropriate range).

Also it is possible to use properties of stencils (like *DataFlow* ∩ *∃ usesProtocol.HTTPProtocol*), so that can be extended to a template language the similar like Microsoft uses (issues might be with negation, cardinality restrictions and other things based on the closed world assumption).

To enable automatic reasoning, it requires to create a threat instance and associate it with a data flow, like:

$$Threat\ (threat1) \quad (5)$$
$$Template1 \subseteq \exists isAffectedBy.\{threat1\}$$

(the “⊆” symbol means “is a subclass of”; and inside “{}” an instance is put).

The “flow” instance (1-2) will be recognized as an instance of the “Template1” concept (4), so according (5) “flow” is affected by “threat1”, and “threat1” affects “flow” (as “affects” is reverse to “isAffectedBy”).

By creation flow templates like (4) and descriptions of threats like (5) it is possible to form a model of threats. Then if that model was combined with a set of instances (processes, flows, boundaries, external entities, data stores etc.), representing a DFD diagram, relevant threat instances would be reasoned as the “isAffectedBy” properties of the flow instances, and the affected flows as the “affects” properties of the threats.

To manage knowledge about countermeasures, the “Countermeasure” concept and “isProtectedBy” property are used. If apply a countermeasure instance and connect it to a flow instance, like:

Countermeasure (count1) (6)
Template1 $\subseteq$ $\exists$ *isProtectedBy.{count1}*

it enables the automatic reasoning of possible countermeasures, similar to threats.

Other way to identify countermeasures is to map them to threats, like:

Countermeasure (count2) (7)
mitigates (count2, threat1)

For more precise classification and labelling of threats and countermeasures a set of security objectives (SO) or the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) model can be used. The difference between the SO and STRIDE approaches is that STRIDE tells about the malicious goals, i.e. represents an adversary point of view rather a viewpoint of resource owner, as security objectives. A list of the security objectives used here includes: Confidentiality, Integrity, and Availability (the CIA triad) and extra objectives like Authentication, Non-Repudiation, and Authorization.

To involve the labelling of threats and countermeasures the "labelsSO" and "labelsSTRIDE" properties can be used, like:

labelsSO(count1, SO_Availability) (8)
labelsSTRIDE (threat1, STRIDE_Denial_of_Service)

VI. Building domain-specific threat model

Creation of domain-specific threat model requires two types of activities:

-Extension of the base hierarchy of concepts by domain-specific concepts;

-Definition of threats and mitigations via data flow templates, as the examples (4-5) show, and applying the labels, as the example (8) shows.

To illustrate the building process of a domain-specific threat model we use the CCCTM model, previously adopted as a template for the Microsoft TM tool (github.com/nets4geeks/CCCTM_template). Full description of CCCTM is out of scope of this work (to get details see the link above). Ontology-based implementation of CCCTM, depicted here, is accessible from GitHub too (github.com/nets4geeks/OdTM) as the *OdTMCCCTM.owl* file. Note, both of the CCCTM implementations do not have a mitigation hierarchy, so this aspect is missed here.

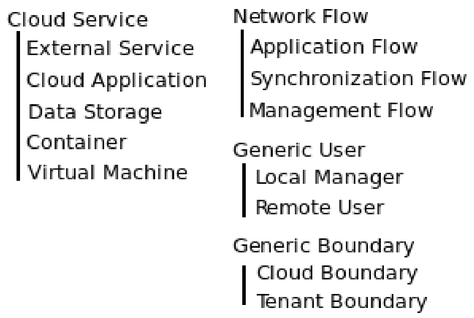


Fig. 6. Components of CCCTM model

The hierarchy of the original CCCTM components is shown in Fig. 6. Integration of this hierarchy to the base ontology can be done by different approaches. One way might be to break the existent hierarchy up with mapping to the concepts of the base threat model, like:

CloudApplication $\subseteq$ *Process* (9)
Container $\subseteq$ *CloudApplication*
VirtualMachine $\subseteq$ *CloudApplication*

where "Process" is taken from the base model, and:

ExternalService $\subseteq$ *ExternalInteractor* (10)
GenericUser $\subseteq$ *ExternalInteractor*
LocalManager $\subseteq$ *GenericUser*
RemoteUser $\subseteq$ *GenericUser*

where "ExternalInteractor" is taken from the base model, and:

ApplicationFlow $\subseteq$ *DataFlow* (11)
ManagementFlow $\subseteq$ *DataFlow*
SynchronizationFlow $\subseteq$ *DataFlow*

where "DataFlow" is taken from the base model, and:

CloudBoundary $\subseteq$ *TrustBorderBoundary* (12)
TenantBoundary $\subseteq$ *TrustBorderBoundary*

where "TrustBorderBoundary" is taken from the base model.

Also it might be possible to apply properties to derived concepts. For example, a cloud application can be characterized by the deployment model (public, private, community, hybrid etc.) and service layer (IaaS, PaaS, SaaS).

An implementation of the deployment model in OWL might be done with applying the "hasCloudModel" property. Its domain is supposed to be "CloudApplication", and range as the "CloudModel" concept, like:

CloudModel $\equiv$ *{ publicCloudModel, privateCloudModel, communityCloudModel, hybridCloudModel }* (13)

An implementation of the service layer with OWL might be done by applying the "hasServiceLayer" property. Its domain is supposed to be "CloudApplication", and range as the "ServiceLayer" concept, like:

ServiceLayer $\equiv$ *{ IaaSServiceLayer, PaaSServiceLayer, SaaSServiceLayer }* (14)

Using (13) and (14) you can define the "VirtualMachine" and "PrivateCloudVirtualMachine" concepts like:

VirtualMachine $\equiv$ *CloudApplication* $\cap$ $\exists$ *hasServiceLayer.{IaaSServiceLayer}* (15)

PrivateCloudVirtualMachine $\equiv$ *VirtualMachine* $\cap$ $\exists$ *hasCloudModel.{privateCloudModel}* (16)

The next step is the definition of threats via data flow templates. There are nine categories of the threats in CCCTM (problems of using External Services, problems of using Cloud Applications, threats to Cloud Applications, management problems of Cloud Applications, organizational problems of Cloud Applications, network flow problems, Management Flow problems, Synchronization Flow problems, Data Flow problems).

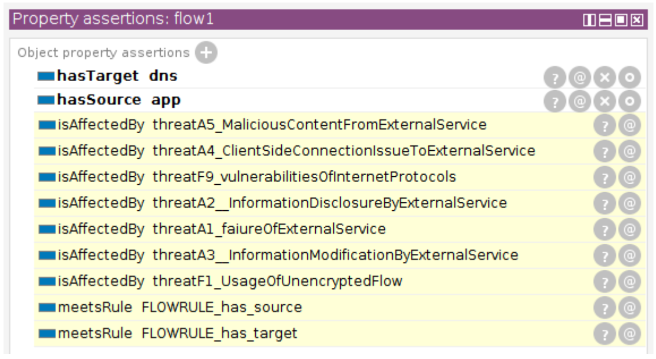
For example, to match threats related to usage of cloud applications (i.e. going from any source to the "CloudApplication") a template should be used like:

$$\begin{gathered}\mathrm{TemplateC} \equiv \mathit{ApplicationFlow} \cap \backslash \\ \exists \mathit{hasTarget.CloudApplication} \\ \mathit{Threat\ (threat\_failure\_of\_application\ )} \backslash \\ \mathit{TemplateC} \sqsubseteq \ \backslash \\ \exists\, \mathit{isAffectedBy.\{threat\_failure\_of\_application\}}\end{gathered} \tag{17}$$

VII. Implementation

You can download the ontology-driven base threat model and CCCTM ontologies from GitHub and using the Protege ontology editor (protege.stanford.edu) apply to the CCCTM ontology a simple DFD description (e.g. include an external DNS service, cloud application and remote user) like:

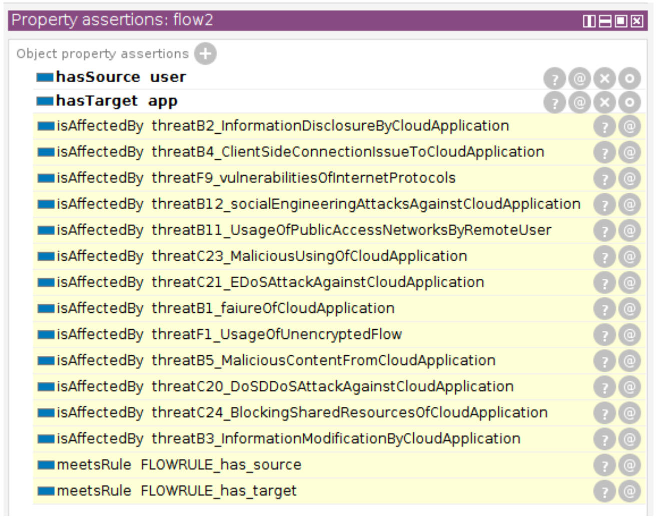
$$\begin{gathered}\mathit{ExternalService\ (dns)} \\ \mathit{CloudService\ (app)} \\ \mathit{RemoteUser\ (user)} \\ \mathit{ApplicationFlow\ (flow1)} \\ \mathit{hasSource\ (flow1,\ app)} \\ \mathit{hasTarget\ (flow1,\ dns)} \\ \mathit{ApplicationFlow\ (flow2)} \\ \mathit{hasSource\ (flow2,\ user)} \\ \mathit{hasTarget\ (flow2,\ app)}\end{gathered} \tag{18}$$


Property assertions: flow1

Object property assertions

hasTarget dns
hasSource app
isAffectedBy threatA5_MaliciousContentFromExternalService
isAffectedBy threatA4_ClientSideConnectionIssueToExternalService
isAffectedBy threatF9_vulnerabilitiesOfInternetProtocols
isAffectedBy threatA2__InformationDisclosureByExternalService
isAffectedBy threatA1_faiureOfExternalService
isAffectedBy threatA3__InformationModificationByExternalService
isAffectedBy threatF1_UsageOfUnencryptedFlow
meetsRule FLOWRULE_has_source
meetsRule FLOWRULE_has_target

Fig. 7. Threats of "flow1" in Protege


Property assertions: flow2

Object property assertions

hasSource user
hasTarget app
isAffectedBy threatB2_InformationDisclosureByCloudApplication
isAffectedBy threatB4_ClientSideConnectionIssueToCloudApplication
isAffectedBy threatF9_vulnerabilitiesOfInternetProtocols
isAffectedBy threatB12_socialEngineeringAttacksAgainstCloudApplication
isAffectedBy threatB11_UsageOfPublicAccessNetworksByRemoteUser
isAffectedBy threatC23_MaliciousUsingOfCloudApplication
isAffectedBy threatC21_EDoSAttackAgainstCloudApplication
isAffectedBy threatB1_faiureOfCloudApplication
isAffectedBy threatF1_UsageOfUnencryptedFlow
isAffectedBy threatB5_MaliciousContentFromCloudApplication
isAffectedBy threatC20_DoSDDoSAttackAgainstCloudApplication
isAffectedBy threatC24_BlockingSharedResourcesOfCloudApplication
isAffectedBy threatB3_InformationModificationByCloudApplication
meetsRule FLOWRULE_has_source
meetsRule FLOWRULE_has_target

Fig. 8. Threats of "flow2" in Protege

After performing the reasoning procedure, you have got the threat list for the flow1 (Fig. 7) and flow2 (Fig. 8) instances. That shows a possibility to apply the ontology-driven approach to the threat modelling process, and receive the similar results to existent means (see Fig. 1-2).

VIII. Conclusions

Our ontology-driven threat modelling approach allows to use graphical notation of DFD diagrams and semantic models to build threat models for modern computer systems. It enables creation of semantic models of different domain-specific areas of IT with well-formed hierarchies of components, threats and mitigations. We have used the description logics as mathematical background and the OWL language as its implementation; that allows to automate the diagram analysis process and reasoning of relevant threats and mitigations.

However, given examples, implemented with Protege, should be considered as proof of concept. For example, advanced implementation of this approach might require to divide namespaces of components and threats for domain-specific threat model (i.e. to create different ontologies for components and threats) to make the hierarchy generation process easier.

In general, a production implementation of the proposed approach requires solving two challenges:

-Automation of creation of domain-specific threat models based on knowledge mining from traditional data sources and integration with existent semantic data sources.

-Creation of software means for building DFD diagrams and their analysis based on domain-specific threat models.

In order to resolve the first challenge, we have researched the problem of extracting and usage knowledge of public directories of software attacks, vulnerabilities, weaknesses to build semantic threat models. In particular, we have built the semantic model (OWL ontology) [8] that is based on the attack pattern (CAPEC - Common Attack Pattern Enumeration and Classification) and weakness (CWE - Common Weakness Enumeration) concepts, and can group (classify) security concepts according given criteria. The work [9] has discussed integration of the DBpedia dataset with the vulnerability catalogue NVD (National Vulnerability Database). The entities (software products and vendors), obtained from CPE (Common Platform Enumeration), have been mapped with the corresponding elements of DBpedia through the DBpedia Spotlight service. NVD uses the CPE entities as a naming scheme for software products, so the semantic model allows to identify NVD records, related to software products, mentioned in DBpedia. The main task for future research is to find a way of mapping attacks and weaknesses (also vulnerabilities as instances of weakness type) to corresponding components of ontology-driven threat models.

Also it is the actual challenge to develop a software means for visual threat modelling. Its features should include ability to extract component hierarchy and their properties from ontology of domain-specific threat model to build appropriate library of DFD components. Also it requires a DFD editor, which allows import/export of DFD diagrams to common XML format; its backend should be able to map graphical notation of DFD diagrams to DL facts (instances of ontology and their properties) and automatically infer threats and mitigations.

References

[1] Abi-Antoun M., Wang D., Torr P. Checking threat modeling data flow diagrams for implementation conformance and security //Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering. – ACM, 2007. – C. 393-396.

[2] Ghijsen, M. A semantic-web approach for modeling computing infrastructures. / M. Ghijsen [et al.] // Computers & Electrical Engineering, - 2013. - 39(8). - P. 2553-2565.

[3] Tasch M. et al. Security analysis of security applications for software defined networks //Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference. – ACM, 2014. – C. 23.

[4] Abomhara M., Gerdes M., Køien G. M. A stride-based threat model for telehealth systems //Norsk informasjonssikkerhetskonferanse (NISK). – 2015. – Т. 8. – №. 1. – С. 82-96.

[5] Cagnazzo M. et al. Threat modeling for mobile health systems //2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). – IEEE, 2018. – С. 314-319.

[6] Sion L. et al. Solution-aware data flow diagrams for security threat modeling //Proceedings of the 33rd Annual ACM Symposium on Applied Computing. – ACM, 2018. – С. 1425-1432.

[7] Berger B. J., Sohr K., Koschke R. Automatically extracting threats from extended data flow diagrams //International Symposium on Engineering Secure Software and Systems. – Springer, Cham, 2016. – С. 56-71.

[8] Brazhuk A. Semantic model of attacks and vulnerabilities based on CAPEC and CWE dictionaries //International Journal of Open Information Technologies. – 2019. – Т. 7. – №. 3. – С. 38-41.

[9] Brazhuk A. Building annotated semantic model of software products towards integration of DBpedia with NVD vulnerability dataset //International Journal of Open Information Technologies. – 2019. – Т. 7. – №. 7. – С. 35-41.

Andrei BRAZHUK
Researcher, senior lecturer at the Yanka Kupala State University of Grodno (Belarus); the PhD student (system analysis) at Belarusian State University of Informatics and Radioelectronics (Minsk, Belarus).

Evgeny OLIZAROVICH
PhD in Engineering sciences, Associate Professor, Head of the Information and Analytical Center, Yanka Kupala State University of Grodno (Belarus).