# Security patterns based approach to automatically select mitigations in ontology-driven threat modelling

Andrei Brazhuk
*Yanka Kupala State University of Grodno*
Grodno, Belarus
brazhuk@grsu.by

***Abstract*—Common approach of the threat modelling includes an analysis of computer system architecture on early stages of development process and creation of threat model. Data Flow Diagrams (DFD) are often used to represent the system organization. The main challenge with threat modelling is that there are no formal approaches to describe the computer system architecture and structured knowledge sources of threats and countermeasures.**

**To overcome these restrictions we have created ontology-driven threat modelling (OdTM) framework based on base threat model, used to develop various domain-specific threat models. Each domain-specific threat model contains a set of typical components of some architectural domain, threats and countermeasures. A system architect describes its computer system with DFD diagram; then automatic reasoning procedures are used to semantically interpret the diagram and figure out relevant threats and countermeasures for the system.**

**We approach a conception of context security patterns as countermeasures. Context security pattern contains a precise description of security problem and its solution. Also it has criteria that allow to automatically map it to system component. We propose three ways to integrate context security patterns with domain-specific threat models: with data flow templates, through association with threats, and the use of labels.**

**All the models, discussed in this work, can be implemented as OWL (Web Ontology Language) ontologies with Description logics (DL) as a mathematical background.**

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

## I. Introduction

Threat modelling is a process of identification of security threats and their countermeasures in order to increase security level of computer system. Common approach of the threat modelling includes two stages. Firstly, an analysis of computer system organization (i.e. its architecture) happens on the early stages of its development process (requirements, design, redesign). Secondly, they build the threat model that represents all security aspects of the system.

Different informal graphical representations of system architecture are used during this process, more often Data Flow Diagrams (DFD) [1]. DFD consists of the objects (stencils) and data flows between these stencils; also a special type of entities exists to group objects and define trust boundaries between them (Figure 1 shows an example of DFD from cloud computing field). Using such diagrams, development team can build an informal threat model of computer system, i.e. figure out possible threats to the system and their countermeasures, through discussions, making notes, and evaluations.

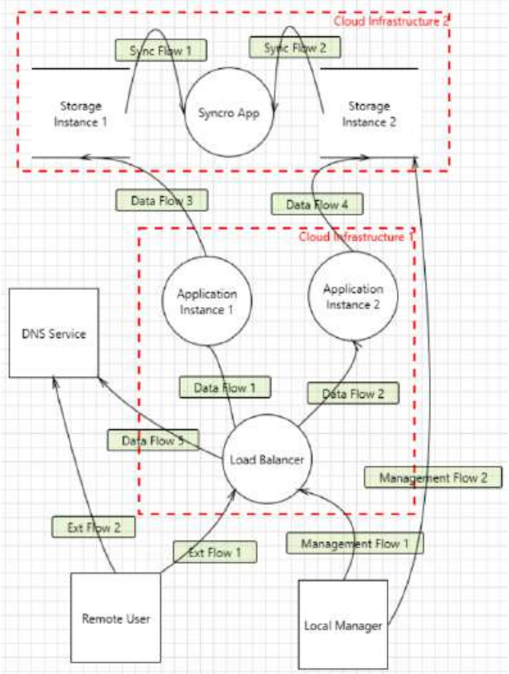


Figure 1. Example of DFD

The main challenge with the threat modelling is that it is too hard to employ formalization and automation there. There is a lack both of formal approaches to describe the computer system architecture, and structured knowledge sources of threats and their countermeasures.

In order to bring a formal approach to this filed [2], we have created *ontology-driven threat modelling (OdTM) framework*. The OdTM framework includes a common approach of the architectural security analysis, method of semantic interpretation of DFD, and automatic reasoning procedures of relevant threats and countermeasures. Our approach is based on the *base threat model* that enables creation of various *domain-specific threat models*. Each domain-specific threat model holds a set of typical components of some architectural domain, threats and countermeasures (security patterns) associated with these components. System architect can describe computer system in terms of a domain-specific model with diagram(s). Then the automatic reasoning procedures can be used to build threat model of the system.

All the models, proposed in this work, can be implemented as OWL (Web Ontology Language) ontologies. OWL has description logics (DL) as a mathematical background. The DL means are able to describe concepts of a domain and relations between them in very formal way and apply automatic reasoning features with relatively low computational complexity.

Another challenge, discussed in this work, refers to employing an approach to choose the mitigations based on security patterns. Security patterns are known as descriptions of security problems that appear in specific contexts and present well proven solution for them [3]. They are created by security experts and represent best security practices for inexpert computer system architects.

We approach a conception of *context security patterns*. Context security pattern is a security pattern that has been placed into a context, i.e. contains more precise interpretation, directly applicable for a system component, rather then generic descriptions of problem and solution. Also, it has criteria that allow to automatically map it to a system component. We propose three ways to integrate context security patterns with domain-specific threat models: with data flow templates, association with threats, and the use of labels.

Also, a problem of formalization of domain specific knowledge is discussed in this work.

## II. Ontology-driven threat modelling framework

The Ontology-driven threat modelling (OdTM) framework is aimed to employ formalization and automation to architectural security analysis of computer systems. It consists of the ontology-driven models and methods that enable automatic analysis of DFD diagrams and building particular threat models. Figure 2 shows structure of the OdTM framework.

The base threat model, implemented as OWL ontology, enables the automatic reasoning features and contains basic concepts and individuals, representing components of the diagrams, threats, countermeasures, and their properties.

To involve the threat modelling of a particular type (domain) of computer systems (e.g. Cloud computing, Fog/Edge computing, SaaS, IaaS, or Software Defined Networks), it requires the following steps:

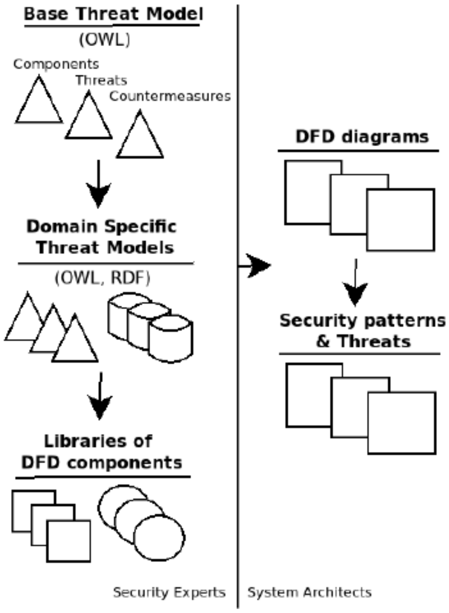


Figure 2. Structure of OdTM framework.

- *Building domain-specific threat model*. To build the model, security experts should extend the base model with specific components, threats and countermeasures, related to the domain. The domain-specific threat model can be considered as a meta model, which depicts the security aspects of this type of computer systems. They are represented as OWL ontologies and can be connected to various external linked data sources (OWL ontologies and RDF data sets).
- *Building domain-specific library of DFD components*. Background procedures automatically extract the component hierarchy from the domain-specific threat model and create the library of DFD stencils, which can be used to draw DFD diagrams.

To create a threat model of a particular computer system, it requires the following steps:

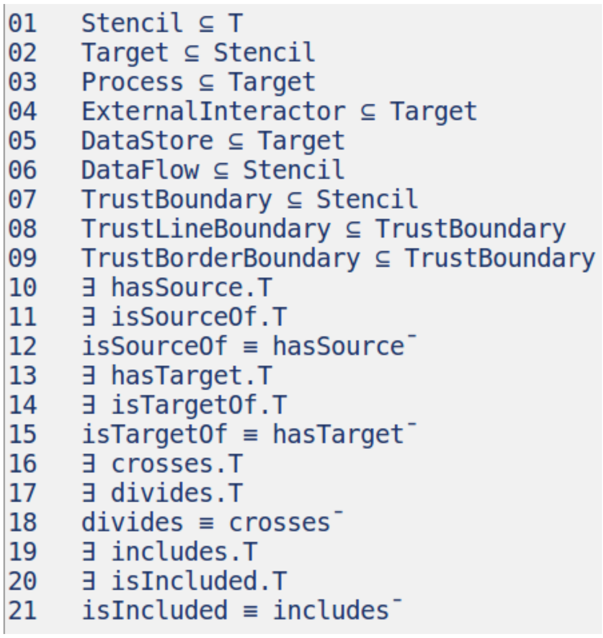
- *Depiction of an architecture of the system as DFD*. A system architect draws a structure of its computer system as the DFD diagram (or the set of diagrams), using the stencils of DFD component library.
- *Semantic interpretation of the DFD*. The background procedures automatically interpret the diagram as a set of semantic instances (components, data flow, boundaries and relations between them) and combine this set with the domain-specific threat model.
- *Automatic reasoning of relevant threats and countermeasures*. Automatic reasoning procedures infer relevant threats and countermeasures from the

semantic interpretation and domain-specific threat model. This allows the background procedures to build lists of the threats and countermeasures for the system.

## III. Semantic interpretation of data flow diagrams

Common ontology description with the DL means uses separation of axioms to the TBox (concepts and properties) and ABox (individuals and their relations) parts. Also, it is supposed that automatic reasoning procedures exist, which allow to get extra facts from an ontology (inferred axioms).

The OdTM base threat model, implemented as OWL ontology, enables semantic interpretation of the diagrams and automatic reasoning of threats and countermeasures. To model a DFD diagram we use a set of concepts (classes) and their properties, described by Figure 3 as DL axioms.



```
01   Stencil ⊆ T
02   Target ⊆ Stencil
03   Process ⊆ Target
04   ExternalInteractor ⊆ Target
05   DataStore ⊆ Target
06   DataFlow ⊆ Stencil
07   TrustBoundary ⊆ Stencil
08   TrustLineBoundary ⊆ TrustBoundary
09   TrustBorderBoundary ⊆ TrustBoundary
10   ∃ hasSource.T
11   ∃ isSourceOf.T
12   isSourceOf ≡ hasSource¯
13   ∃ hasTarget.T
14   ∃ isTargetOf.T
15   isTargetOf ≡ hasTarget¯
16   ∃ crosses.T
17   ∃ divides.T
18   divides ≡ crosses¯
19   ∃ includes.T
20   ∃ isIncluded.T
21   isIncluded ≡ includes¯
```

Figure 3. Semantic interpretation of DFD diagram (a part of TBox).

Axioms (Ax.) 01-09 in Figure 3 model the hierarchy of the DFD base stencils. Three main concepts are derived from the "Stencil" concept: "Targets", "TrustBoundaries", and "DataFlows".

An instance of the "DataFlow" concept represents a directional flow from a source "Target" instance to a target "Target" instance. To model this, a data flow should have two properties: "hasSource" (Ax. 10) and "hasTarget" (Ax. 13); for both of them a range is supposed to be Target. The properties "isSourceOf" (Ax. 11) and "isTargetOf" (Ax. 14) are inverse to previous two ones (Ax. 12, 15) and allow to infer that a target is a starting edge or ending edge of some data flow.

A data flow can cross some instance of the "TrustLineBoundary"; to tell this, the "crosses" property is used (Ax. 16). Also, a line boundary might divide some data flow; to describe this, the "divides" property is used (Ax. 17). The last two properties are inverse (Ax. 18).

A target might be included into some "TrustBorderBoundary" instance; to model this, the "isIncluded" property is used (Ax. 20). Also, a TrustBorderBoundary instance contains, or "includes" (Ax. 19) some targets. The "include" and "isIncluded" properties are reverse (Ax. 21).

To semantically model a diagram that describes an architecture of a computer system, we should apply the ABox axioms to the ontology. This includes instances of the base concepts (or derived of them) and their properties: "Targets", "DataFlows" with "hasSource" and "hasTarget", "TrustLineBoundaries "with "crosses", "TrustBorderBoundaries" with "includes".

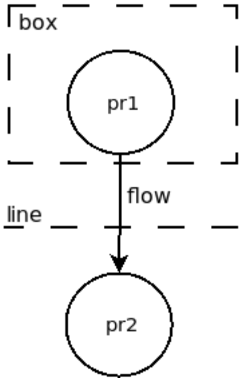


Figure 4. A simple DFD diagram.

Figure 4 depicts a simple example of DFD, and below is shown a possible semantic interpretation of the diagram:

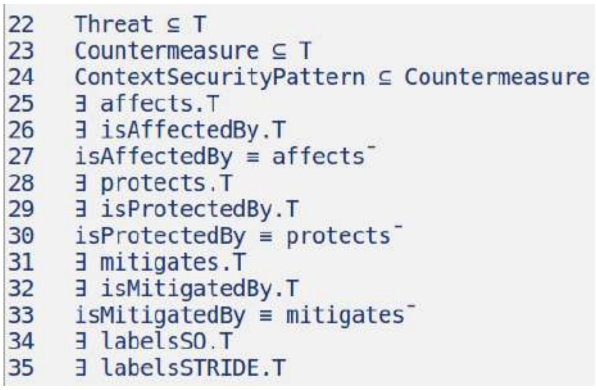
$Process(pr1)$
$Process(pr2)$
$DataFlow(flow)$
$hasSource(flow, pr1)$
$hasTarget(flow, pr2)$
$TrustLineBoundary(line)$
$crosses(flow, line)$
$TrustBorderBoundary(box)$
$includes(box, pr1)$

Note, the automatic reasoning procedures would be able to infer additional facts from the ABox sets like this. We do not need to tell in this example, that "pr1" is a source of "flow", "pr2" is its target, "line" divides "flow", "pr1" is included into "box". These facts would be inferred by the automatic reasoning procedures (see Ax. 12, 15, 18, 21).

## IV. Automatic selection of security patterns

In modern computer systems data flows are origins of security issues, because most of the attacks are remote and sourced from the local and remote networks. Usually threats are applied to computer system by data flows. Also, reasons for adding a countermeasure (security pattern) to particular architecture depend on the presence of a data flow.

To model threats and countermeasures we use a set of concepts and their properties, described by Figure 5.



```
22   Threat ⊑ T
23   Countermeasure ⊑ T
24   ContextSecurityPattern ⊑ Countermeasure
25   ∃ affects.T
26   ∃ isAffectedBy.T
27   isAffectedBy ≡ affects⁻
28   ∃ protects.T
29   ∃ isProtectedBy.T
30   isProtectedBy ≡ protects⁻
31   ∃ mitigates.T
32   ∃ isMitigatedBy.T
33   isMitigatedBy ≡ mitigates⁻
34   ∃ labelsSO.T
35   ∃ labelsSTRIDE.T
```

Figure 5. Countermeasures and threats (a part of TBox).

An instance of the “Threat” concept (Ax. 22) “affects” some data flow. Also the inverse property called “isAffectedBy” is used (Ax. 26, 27).

An instance of the “Countermeasure” concept “protects” (Ax. 28) some data flow, and a data flow “isProtected by a countermeasure (Ax. 29, 30). Also, countermeasure “mitigates” (Ax. 31) some threat, and a threat “isMitigated” by some countermeasure (Ax. 32, 33).

Security patterns in the model are extended to the “ContextSecurityPattern” concepts. They are derived concepts of “Countermeasures” (Ax. 24). The base threat model has three ways to enable the automatic reasoning of context security patterns, as well as other kinds of mitigations, through countermeasures: A) data flow templates, B) association with threats, C) the use of labels.

A) *The first option is the use of data flow templates*. A flow template should be defined as a concept with the “hasSource”, “hasTarget”, “crosses” (and other) properties like:

$$Template1 \equiv DataFlow$$
$$\sqcap \exists hasSource.Process$$
$$\sqcap \exists hasTarget.Process$$
$$\sqcap \exists crosses.TrustLineBoundary$$

To enable automatic reasoning, it requires to create a instance of context security pattern and associate it with a data flow template, like:

$$ContextSecurityPattern(pattern1)$$
$$Template1 \subseteq \exists isProtectedBy.\{pattern1\}$$

Using last three axioms, the “flow” instance (from Figure 4) would be recognized by the automatic reasoning procedures as an instance of the “Template1” concept, so “flow” would be protected by “pattern1”, and “pattern1” would protect “flow”.

By creation flow templates and descriptions of patterns, like shown above, it is possible to form a model of context security patterns. The same way it can be possible to create a model of threats by mapping the Threat instances to data flow patterns with the “isAffected” property.

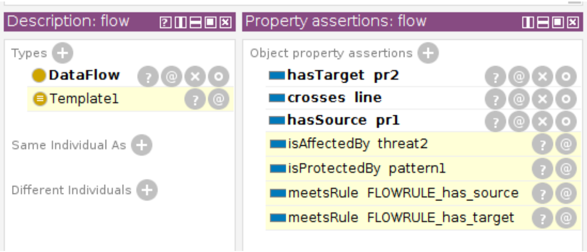
B) *The next way to employ the context security patterns is to map them to threats with the “mitigates” property*, like:

$$Threat(threat2)$$
$$Template1 \subseteq \exists isAffectedBy.\{threat2\}$$
$$ContextSecurityPattern(pattern2)$$
$$mitigates(pattern2, threat2)$$

C) *For more precise classification of threats and countermeasures a set of security objectives (SO) and the STRIDE model can be used*. STRIDE stands from Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. 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.

Using the “labelsSO” and “labelsSTRIDE” properties (Ax. 34-35), it is possible to label threats, countermeasures and context security patterns:

$$labelsSO(pattern2, SO_{A}vailability)$$
$$labelsSTRIDE(threat2, STRIDE_{D}enial_{o}f_{S}ervice)$$



Description: flow

Types

DataFlow

Template1

Same Individual As

Different Individuals

Property assertions: flow

Object property assertions

hasTarget pr2

crosses line

hasSource pr1

isAffectedBy threat2

isProtectedBy pattern1

meetsRule FLOWRULE_has_source

meetsRule FLOWRULE_has_target

Figure 6. Automatic reasoning example in Protege.

It would be easy to implement the discussed above model (Figures 3 and 5) as OWL ontology and, using a reasoner (like HermiT, Fact++, or Pellet), check the feasibility of proposed ideas.

Our implementation of the OdTM base threat model has freely been published with the GitHub service (https://github.com/nets4geeks/OdTM) as the *OdTM-BaseThreatModel.owl* file.

Figure 6 shows the properties of the "flow" instance from Figure 4 after the automatic reasoning performed with Protege.

## V. Building domain-specific threat models

Figure 7 shows the process of the domain specific knowledge formalization, used to build domain-specific threat models.

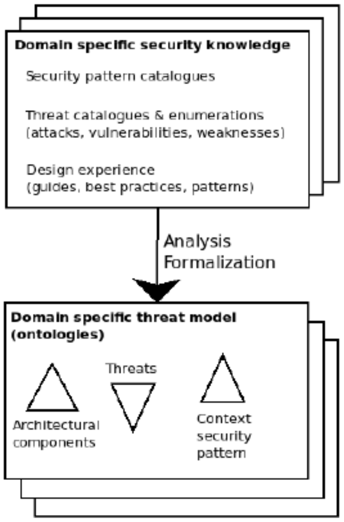

Domain specific security knowledge
Security pattern catalogues
Threat catalogues & enumerations (attacks, vulnerabilities, weaknesses)
Design experience (guides, best practices, patterns)
Analysis Formalization
Domain specific threat model (ontologies)
Threats
Architectural components
Context security pattern

Figure 7. Formalization of domain specific knowledge.

Steps to build a domain-specific threat model are following:

- *Create a sub model of architectural components.* That includes findings all the items (components, relations, boundaries) that a computer system of particular domain can have. This enumeration should be used to extend the component hierarchy of the base threat model with domain specific concepts.
- *Create a sub model of threats.* This includes an enumeration creation of possible domain specific threats, giving proper definitions for the threats, mapping them to data flows and patterns. To make easier the threat analysis, it is possible to build additional semantic models. In particular, we have built the OWL ontology [4] that is based on the attack pattern (CAPEC - Common Attack Pattern Enumeration and Classification) and weakness (CWE - Common Weakness Enumeration) concepts, and it is able to classify security concepts according given criteria. To process raw sources of vulnerability information, like NVD (National Vulnerability Database), it strongly requires implying various NLP (natural language processing) methods [5].
- *Create a sub model of context security patterns.* This includes an enumeration creation of context security patterns and mapping them to data flows and threats. However, well-known security patterns are textual descriptions of security problems, made in some format in technology-independent way, e.g. the POSA (Pattern Oriented Software Architecture) templates. Security experts are able to understand generic descriptions of security patterns and employ them as design decisions to specific computer systems. But applying them as automatically inferred solutions (i.e. putting in a context) requires some extra adaptation steps. It can be argued about two hundred common security patterns exist [6], and there are hardly any sources able to create patterns automatically. So, manual and semi automatic methods of creation of context security patterns are preferred for this kind of job.
- *Label the threats and context security patterns to the security objectives and STRIDE items.*

## VI. Related work

Works [6] and [7] have presented an ontological approach to manage security patterns. The ontologies facilitates mapping between the context aspects and security patterns themselves, and therefore enables automatic pattern selection during the secure system development process.

The most powerful effort for the threat modelling automation has made by Microsoft with the Threat Modelling (TM) software. Microsoft TM consists of a drag-and-drop DFD editor, simple rule-based reasoner, report subsystem, and built-in threat template editor. Microsoft uses a simple rule language to describe associations threats with data flows. The XML format is used to save threat templates.

Some works have used the Microsoft approach for research and creation of security threat models based on DFD [8], [9], [10]. Hovewer, there are some issues with the Microsoft implementation. The Microsoft tool only operates with two level hierarchy of objects (stencils and derived stencils) and threats (categories and threat types), however for description of complex computer systems and their threats it usually requires more layers of abstraction. Also there is a lack of full-featured countermeasure hierarchy, which would allow a user to choose a countermeasure to a threat from a relevant list. Our work has gone to find a way to overcome these restrictions with creation of semantic models with well-formed hierarchies of components, threats and countermeasures.

Work [11] 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 (their knowledge base uses a domain-specific rule language, based on a graph query language), and creation of a visual EDFD viewer. Work [12] has proposed very similar findings to what our research has offered. They have researched a challenge of automatic correction of security issues in declarative deployment models of cloud services based on the ontological approach and security patterns. It can be argued that our OdTM approach conceptually satisfies their topology-based deployment meta model. However, their implementation is directly based on First-order logic (FOL) and the low-level logical programming (Prolog).

An advantage of our approach to compare with other works [11] and [12] is the use of Description logics (DL) through OWL and the automatic reasoning features as an implementation. This allows to employ an object-oriented approach to the knowledge management system design, i.e. provide better representation for users, stricter formalization, and easier ways to implement. Also it is possible to apply (if necessary) various high-level means, like the SWRL (Semantic Web Rule Language) rules, the SPARQL (SPARQL Protocol and RDF Query Language) queries. And an implementation based on OWL enables integration with linked open data (LOD) sources.

## VII. Conclusions

The OdTM framework is based on domain-specific threat models with appropriate libraries of DFD components. Each domain-specific threat model is a meta model of threats of particular computer system domain, represented as the DL compatible formalization. It contains axioms that can be considered as a TBox (mixed with the instances of threats and countermeasures). To model a DFD diagram that describes an architecture of a particular computer system, it requires to interpret the diagram as a set of instances (ABox). Using these ABox and mixed TBox, the automatic reasoning procedures can infer relevant threats and countermeasures for the computer system.

Informally, to build domain-specific threat it is necessary to extend the base threat model by creation of a hierarchy of domain typical components, association of threats and context security patterns to data flow templates, mapping threats and patterns to each other, and labeling them by the security objectives and STRIDE items. However, formalization of domain specific security knowledge should be considered as its transformation to sub models of architectural components, threats and context security patterns. In future research we are going to learn these processes in order to create methods and models that enable (semi) automatic building of domain-specific threat models.

A special field of interest there would be transformation of general security patterns to context security patterns. A system architect thinks of a computer system in domain specific terminology and expects that proposed solutions would be described the same way. Also, criteria to map or not a pattern to a system design are quite important.

## References

[1] M. Abi-Antoun, D. Wang, P.Torr, "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, pp. 393-396, 2007.

[2] E.V. Olizarovich, A.I. Brazhuk "Kontseptual'nye osnovy analiza modelei informatsionnoi bezopasnosti oblachnykh sistem klassa «infrastruktura kak usluga»" [Conceptual framework of analysis of information security models of cloud systems of the class «Infrastructure as a Service»], Doklady BGUIR, 2019. vol. 6(124), pp. 12-20.

[3] M. Schumacher, et al. Security Patterns: Integrating security and systems engineering. John Wiley and Sons, 2013.

[4] A. Brazhuk "Semantic model of attacks and vulnerabilities based on CAPEC and CWE dictionaries," International Journal of Open Information Technologies vol. 7, no. 3, pp. 38-41, 2019.

[5] A. Brazhuk "Building annotated semantic model of software products towards integration of DBpedia with NVD vulnerability dataset." IJOIT, vol. 7, no. 7, pp. 35-41, 2019.

[6] A.P. Vale, E. B. Fernández, "An Ontology for Security Patterns". In 2019 38th International Conference of the Chilean Computer Science Society (SCCC), pp. 1-8, 2019.

[7] H. Guan, H. Yang, J Wang, "An ontology-based approach to security pattern selection." International Journal of Automation and Computing 13.2, pp. 168-182, 2016

[8] M. Tasch, et al. "Security analysis of security applications for software defined networks." Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference. ACM, 2014.

[9] M. Abomhara, M. Gerdes, and G. M. Køien. "A stride-based threat model for telehealth systems." Norsk informasjonssikkerhetskonferanse (NISK) 8.1, p. 82-96, 2015.

[10] L. Sion, et al. "Solution-aware data flow diagrams for security threat modeling." Proceedings of the 33rd Annual ACM Symposium on Applied Computing. ACM, 2018.

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

[12] K.Saatkamp, et al. "An Approach to Determine and Apply Solutions to Solve Detected Problems in Restructured Deployment Models using First-order Logic." SICS Software-Intensive Cyber-Physical Systems, vol, 34, no. 2-3, pp.85-97, 2019.

# Подход на основе шаблонов безопасности для определения контрмер в онтологическом моделировании угроз

Бражук А.И.

Описан подход к моделированию угроз компьютерных систем на основе предметно-ориентированных моделей угроз, который позволяет автоматически определять угрозы и контрмеры по графическому представлению структуры системы в виде диаграми потоков данных. Описана базовая онтологическая модель угроз. Предложена концепция контекстных шаблонов безопасности для определения контрмер. Для реализации предлагается использовать язык онтологий OWL и функции автоматического логического вывода.

Received 26.11.19