Winter 12-11-2022

Configuration and management of security procedures with dedicated 'spa-lang' domain language in security engineering

Tomasz Krym
Lodz University of Technology

Aneta Poniszewska-Marańda
Lodz University of Technology, aneta.poniszewska-maranda@p.lodz.pl

Łukasz Chomątek
Lodz University of Technology

Jakub Chłapiński
Kodegenix

Configuration and management of security procedures with dedicated 'spa-lang' domain language in security engineering

Tomasz Krym
Lodz University of Technology, Kodegenix
Łódź, Poland

Aneta Poniszewska-Marańda[1]
Lodz University of Technology
Łódź, Poland

Łukasz Chomątek
Lodz University of Technology
Łódź, Poland

Jakub Chłapiński
Kodegenix
Łódź, Poland

ABSTRACT

The security policy should contain all the information necessary to make proper security decisions. The rules and needs for specific security measures and methods should be explained in understandable way. None of the existing security mechanisms can guarantee complete protection against threats. In extreme cases, improperly used security mechanisms can lower the level of protection, giving the impression of security that is actually lacking. To enable simple and automated definition of security procedures for IT system of a company or organization, available not only to qualified IT professionals, e.g. system administrators, but also to the company's management staff, it was decided to create an Intelligent System for Automation and Analysis of Security Procedures (iSPA). The paper presents the proposal of use the developed domain language, named 'spa-lang' for configuration and management of security procedures in security system engineering based on BPMN (Business Process Model and Notation) standard.

Keywords: information system security; security policy; security procedures; domain language; BPMN (Business Process Model and Notation).

[1] Corresponding author. aneta.poniszewska-maranda@p.lodz.pl +48 42 631 27 96

INTRODUCTION

Ensuring data security means meeting several requirements related to the properties of a secure information system. A secure system is one that ensures: confidentiality, integrity, availability. The first two aspects relate to data, the third relates to more generally understood system resources. Data confidentiality means a state in which data (e.g. personal data) is not and cannot be disclosed to unauthorized persons. Data integrity is a state in which data is not susceptible to any unauthorized modification. The aspect of accessibility means that authorized persons can use the system resources within the scope of their access rights. The mentioned requirements must be guaranteed during the storage, transmission and processing of data. At each of these stages, different threats may arise and different protection measures are necessary to prevent and detect the violations. The consequences of violating one of the above-mentioned aspects of security may be direct or indirect. The immediate effects of security breaches are reduced productivity of the enterprise and the need to incur expenses to "rebuild" the system. The indirect effects include: decrease in customer confidence in the company/institution, decrease in the value of the company's shares and legal consequences resulting from violations of the law in force in the field of data protection. Costs related to direct effects are relatively easy to quantify, as opposed to those related to indirect effects. The overall costs associated with security breaches can be very high.

None of the existing security mechanisms can guarantee complete protection against threats. There are no fully secure systems. Moreover, the user himself is often unaware that the security mechanism he/she uses is ineffective. In extreme cases, improperly used security mechanisms can lower the level of protection, giving the impression of security that is actually lacking. To enable simple and automated definition of security procedures for the IT system of a

company or organization, available not only to qualified IT professionals, e.g. system administrators, but also to the company's management staff, it was proposed to create an Intelligent System for Automation and Analysis of Security Procedures (iSPA). The main assumptions of the iSPA system are as follows: (1) Improvement and reduction of infrastructure configuration costs through its ergonomic automation, at least in terms of devices and systems configured through the SSH console (later also WinRM, SNMP, etc.). (2) Providing a tool for the staff with weaker technical background to design the logic of security procedures at the interface and integration of various subsystems and services. This goal was achieved by developing a language for defining spa-lang safety procedures, and mechanisms for its representation in the form of BPMN diagrams. (3) Providing mechanisms to verify that the designed security procedures are properly implemented.

In order to justify the need to create a new domain language dedicated to the iSPA system, an analysis of existing solutions in the field of domain languages related to IT security was carried out. The performed analysis showed a significant technical uncertainty in terms of the complexity of domain languages described in the literature. The following main solutions were identified: user behavior patterns (Sharghi and Kamran 2017), security of sensitive data (Dijk et al. 2017), defining rules for alarm in certain cases (Nazir et al. 2017). These works present domain languages that enable modelling of security rules in issues constituting a subset of the scope of security procedures developed by us. Therefore, it was undertaken to build the syntax of domain language and the mechanism of translation into security procedures. It is also important to comply with the currently used standards, e.g. BPMN (Business Process Model and Notation) (Dumas et al. 2013; Chang 2016; Kumar 2018), in order to reduce the barrier of learning a specific language for people who do not have detailed knowledge. Another important

stage in implementation of iSPA system was development of method of intelligent analysis of monitored systems, taking into account the set of security rules provided by the client. This includes specifying a set of criteria to evaluate a security rule violation and providing a reference implementation of a new set of security rule violation evaluation criteria. Works in the area of intelligent security monitoring in the field of IT include, e.g. research on transaction security (Kim et al. 2017), malware attacks (Elsemary 2016), tracking user behavior based on biometric characteristics (Zaghetto et al. 2017), detecting anomalies in user behavior (Gates et al. 2014).

The paper presents the proposal of use the developed domain language, named 'spa-lang' for configuration and management of security procedures in security system engineering. The possibility of graphical representation of procedures in 'spa-lang' language in the form of diagrams compliant with BPMN standard was developed. This feature it allows building and performing complex security procedures also for users with only basic IT competences (Krym et al. 2021). The paper is structured as follows: Section 2 gives the outline of chosen security aspects of information systems and security procedures. Section 3 deals with the proposition of intelligent system for automation and analysis of security procedures while section 4 presents the overview of syntax of 'spa-lang' language created for configuration of security procedures.

SECURITY ASPECTS OF INFORMATION SYSTEMS AND SECURITY PROCEDURES

When developing a security policy for the IT system of a specific institution, a number of factors should be taken into account. The most important are: system objects, threats, acceptable risk, the level of investment in security, the value of information, potential losses resulting from security breaches. The specificity of the institution in which the IT system operates is also important in the development. A policy developed for the needs of one institution should not be

implemented in another institution, even with a similar profile of activity. The security policy should be dynamic, which means that changes taking place in an institution must be reflected in changes in the security policy. It is necessary to take into account the modifications within the institution and those that occur outside, including the transformation of the applicable law, changes in technology, and the emergence of new threats. Security policy must take into account the economic aspect of the issues. It is necessary to find a compromise between the reduction of losses resulting from the activation of the existing threats and the expenses incurred for the introduction and operation of the protection system. Another problem is the fact that over-protection may be an obstacle to the functioning of the IT system, and thus the entire institution. Usually, along with the increase in security, the functionality of the IT system decreases. Safety is achieved by introducing restrictions and performing additional activities that are not "productive", i.e. not profitable. As the development of information systems tends to increase functionality and efficiency, ensuring security becomes more and more difficult.

Due to the basic aspect of security, which is accessibility, security procedures include, among others the configuration of all services provided to users of the ICT infrastructure (e.g. domain internet applications such as financial and accounting systems, CRM, VoIP) and all technical services necessary for the functioning of the infrastructure (e.g. LDAP, DNS, DHCP) (Argyropoulos at al. 2019, Salnitri et al. 2014). Thus, within a given enterprise, security procedures also include the problem of configuring all services to ensure their availability to end users. Automatic security procedures in IT systems can be reduced to the following types of activities: (1) *Configuration of ICT infrastructure, which contains*: (a) Hardware configuration. (b) Configuration of operating systems. (c) Configuration of technical services necessary for the operation of the infrastructure (e.g. DHCP, DNS, LDAP). (d) Configuration of domain services

and applications, made available to end users as part of the company's operations. (e) The use of specialized HSM (High Security Module) devices for storing cryptographic content in the services. (2) *Monitoring events in the infrastructure (at least those related to security)*: (a) Monitoring, aggregation and analysis of logs from individual services (e.g. using Syslog, ELK/ElasticSearch Logstash Kibana). (b) Access control systems (e.g. at the office, in the server room). (c) Fire systems. (d) HVAC systems. (d) UTM/IDS systems for network traffic analysis. (3) *Performing automated actions in response to event*: (a) "Short loop" reactions, implemented mainly through specialized services and devices. (b) "Long loop" reactions, implementing security procedures at higher level, requiring integration between separate systems. (4) *Security audit*: (a) Safe and undeniable collection and storage of security audit information that covers. (b) Verification if security procedures are properly implemented in ICT infrastructure.

SYSTEM FOR AUTOMATION AND ANALYSIS OF SECURITY PROCEDURES

In the framework of our research and development works, the concept and technology of the new security system *iSPA (Intelligent System for Automation and Analysis of Security Procedures)* was developed. The system is intended for distributed IT environments, e.g. Data Center, industrial control systems and the Internet of Things (IoT). The new technology enables the consistent management of the client's organization's business activities and IT security aspects through: (1) *Automation of security management procedures*, enabling the coordination of automatically performed technical activities (tasks of IT administrators), organizational and formal (authorizations, consents, etc.). (2) *Intelligent analysis of monitored systems*, thanks to which individual events from monitored systems will be interpreted in the context of actual threats to the continuity of the client's business activities. (3) *Securing the undeniable*

documentation of taken actions, through the use of a specialized security system, based on dedicated *hardware security module* (HSM) and the PKI certification center.

The essence of iSPA system technology is a new declarative "*spa-lang" (Security Process Automation Language)* describing the features, functions and tasks of all IT resources (hardware, networks, software and network services). Based on this language, the new algorithms were and will be created to enable the *automatic* performance of complex operations modelled in the form of BPMN processes, intelligent analysis and security audit of the monitored distributed IT systems, as well as the presentation of monitoring results in a graphical domain language, easy to interpret for the end user, not necessarily having technical competences for the administration of IT resources. This aspect of the iSPA system is a particularly important differentiator in relation to competing solutions currently available on the global market (e.g. Cisco ACI), as it will enable security supervision directly to the persons responsible for it (e.g. management), without the need for technical staff (Zareen et al. 2020; Maines et al. 2015; Maines et al. 2016; Rodriguez et al. 2007).

In order to ensure the highest level of security of the procedures performed, the system should have a built-in, dedicated hardware security module (HSM). In this way, all activities related to security procedures take place only through the HSM module/modules using the PKI certification system, which will ensure the highest level of access security, non-repudiation and audit. The creation of a dedicated 'spa-lang' domain safety management language within the iSPA system included the following activities: (1) Development of a domain language syntax that eliminates the limitations of previously developed domain languages for modelling issues related to IT security based on BPMN standard. These works were related to: defining the set of language idioms at the level of the field, experimental selection of the scope of terms used in the

language, determining the level of detail of the used linguistic constructions (finding a compromise between the expressiveness of the language and the possibilities of syntax expansion), ergonomics of working with language. (2) Development of a mechanism of translation of 'spa-lang' language records into safety procedures covering the indicated spectrum of issues. These works are related to: estimating the computational overhead resulting from the use of the domain language and translation into security procedures, the determination and levelling of translation barriers due to the syntactic structures of the domain language. (3) Development of an algorithm for translating the 'spa-lang' language into a record of commands enabling the implementation of the desired security rules. (4) Experimental verification of the completeness of the syntax of the domain language 'spa-lang' and the correct operation of the translation mechanism on a set of test use cases including reference safety procedures (laboratory conditions), tests related to areas such as: computational overhead, ergonomics of working with language, restrictions on writing and translation of security procedures in built domain language.

The basis for development of a dedicated 'spa-lang' security management language, the syntax of which is presented in the next section, is a reference set of safety procedures developed together with the Client. It includes the sets of security rules in the field of: (a) managing user accounts (profiles) in Linux, Microsoft Windows 7/8/10, MacOS systems, (b) defining authentication rules for user accounts and services (certificate: digital signature, password, access card, fingerprint reader, face recognition: Windows Hello technologies, face-recognition for Linux/MacOS) for the above-mentioned operating systems, (c) defining the exchange policy and syntax of passwords, (d) defining additional, non-standard authentication methods through support for authentication modules (Pluggable Authentication Module, PAM) for Linux, Windows, MacOS, (e) defining the security policy in the event of repeated unsuccessful login

attempts (e.g. blocking an account for a specified period of time, sending e-mail notifications, text messages), (f) storing credentials in centralized Active Directory/LDAP services, (g) defining a policy for local storage of credentials in the event of no access to the central system (LDAP/Active Directory server), so that users previously authenticated on a given device can log into the operating system even in emergency situations, (h) support for authenticating user and service accounts using the Kerberos protocol, (i) defining user and service access policy to the WLAN network, (j) defining the policy of users and services access to the LAN (VLAN), (k) defining the user and service access policy to the public WAN network, (l), defining the policy of users and services access to network shares of NFS and SMB protocols, (m) defining user and service access policies to local file systems on individual devices, (n) defining security audit rules, (o) configuration of the audit system in terms of location and log storage rules, (p) configuration of SELinux security rules, (r) support for sending automatic notifications via email and SMS, (s) support for reacting (performing procedures) initiated via a sent email or text message (e.g. blocking an account on command sent by SMS from the administrator), while maintaining an appropriate level of authentication for this method of communication.

'SPA-LANG' LANGUAGE FOR CONFIGURATION OF SECURITY PROCEDURES

Spa-lang language is a *domain language*, i.e. it is not a general-purpose language. This language allows to define procedures to be executed sequentially (synchronously), and to generate and react to asynchronous events. The language has a text form and defined translation mechanisms into the form of BPMN diagrams. A text syntax and a separate translation to BPMN diagrams were chosen, because more complicated procedures will be easier to implement in text form, especially for administrators with more programming support, while the graphical form of BPMN is very well suited for introducing relatively simple corrections (e.g. changing required

password length for users in Active Directory domain), especially for non-technical personnel. In BPMN representation, with the help of collapsible subprocesses, it is possible to ergonomically hide implementation details that are irrelevant (incomprehensible) to an ordinary user.

Syntax of 'spa-lang' language

The exemplary program used for description of spa-lang' language syntax is as follows:

```
// This is a comment     /* This is a block comment */     # This is also a line comment
// Every text file can be imported as a string variable
const ADD_USER_SCRIPT = import "./add_user_ldap.sh"
// JSON, YAML and TOML files are imported as object variables
const SOME_DATA = import "./data.json"
// but can be also imported as a string variable
const SOME_DATA_TEXT = import "./data.json", format = text
// module definition
mod ldap {
   // Variables placed at module level consist persisted global stateof
   // module execution
   let status;
   // All used events need to be declared. Can be public or not.
   pub event user_add_ok;
   // This event can be also reacted on or emitted in other modules
   event user_add_error;
   // This event is private, so it can only be emitted and reacted to in this module
   // Code can be decorated anywhere by metadata, used in graphical representation.
   // Metadata are assigned to lexical element directly next to it (procedure below)
   @position { x: 10, y: 10, width: 300, height: 80 }
   proc add_user(user) {    /* ... */    } }
```

The basic rules of the language are as follows: (1) All variables hold data about the type of *Opath* expression. (2) The basic compilation unit is the *module*. Modules may include: (a) one or more *procedures*, (b) definitions of *event* types, (c) variables representing the module's global execution context (available to all procedures in the module). (3) The *procedure* is mapped to a BPMN diagram. It contains code that is executed synchronously (inline procedures such as run_script and conditional statements), and code for reaction to events, executed asynchronously. (4) Procedures may include: (a) by any number of arguments, (b) *if ... else ...*

conditional statements (c) expression *for ... in ...*, (d) calls to stored procedures (run_script(), file_copy()), (e) instruction to emit events (*emit ...*), (f) instructions to react to an event (*on ...*). (5) In addition, code can be annotated anywhere with metadata in the following format: *@ ...*

Atomic operations

Configuration management on hosts comes down to *atomic operations*: (a) editing text files (configuration files) – so-called template engine to populate configuration file templates with infrastructure description data was implemented, (b) parsing text files to read the existing configuration – a library in the Rust language has been implemented for this purpose to automatically generate parsers for arbitrary grammars (context-free or context-free grammars), and entering their content in an object-oriented form into the 'spa-lang' context program, (c) executing console commands and system scripts, (d) efficient file transfer/synchronization through integration with the RSYNC protocol. Atomic operations are syntactically modelled similar to functions in general-purpose languages:

```
proc add_user(user) {
 run_script('./smb_before_create_user.sh')
 run_command('samba-tool create user ${user.username}')
 file_copy('./default_profile/**/*', '/storage/profiles/${user.username}/',
 { user: user })    }
```

Algorithm for mapping atomic operations to BPMN

Atomic operations in the BPMN representation are realized by *Task*. Successive atomic operations in 'spa-lang' code in BPMN representation are linked through *Sequence Flow*. The mapping algorithm for above fragment of 'spa-lang' code is presented in figure 1.

Procedures and modules

The main executable part of the 'spa-lang' program is the *procedure*. A procedure can take any number of named arguments. Arguments, like all variables, are of the *Opath* expression

type. A procedure can contain the sequences of atomic actions, conditional statements, loops, and *on* expressions that mean the continuation of a procedure after the specified event occurs.

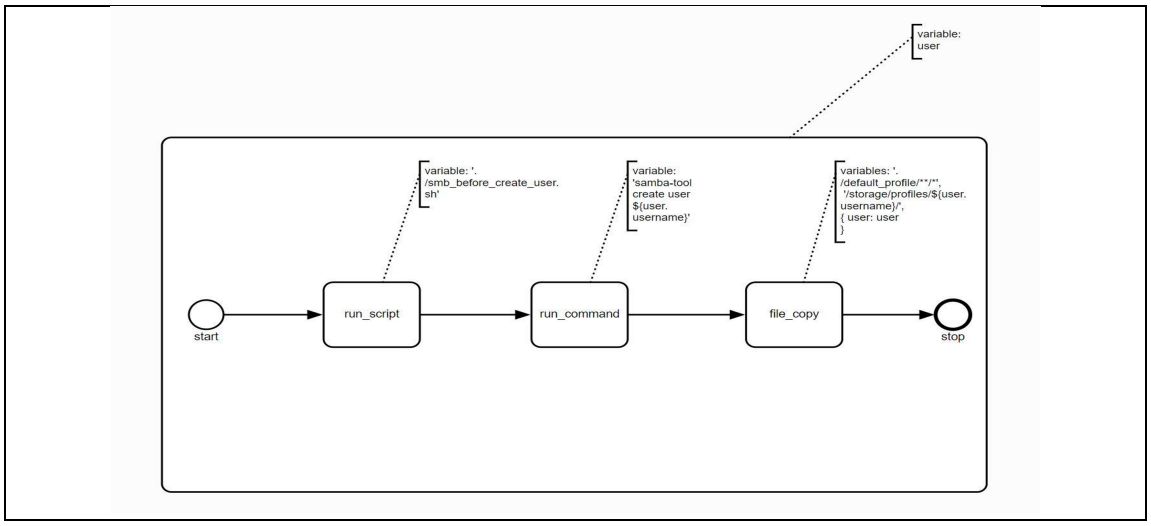

Figure 1. Mapping elementary activities from 'spa-lang' to BPMN diagrams

The compilation unit is the *module*. A valid program in 'spa-lang' language must contain at least one module. A module can contain *procedures*, definitions of *events* generated by procedures inside the module, and *event service procedures (handlers)* executed when responding to the event. Handlers differ from procedures in that they cannot have arguments, they are always executed in the context of a machine on which the event was generated. While in its body it is possible to access the event object (or its fields through destructuring, as in the listing below), which handler is executing. The module may also contain global state variables.

```
mod active_directory {
  @default_host: 'ad01.example.org'
    proc add_user(user) {
      run_script('./smb_before_create_user.sh')
      run_command('samba-tool create user ${user.username}')
      file_copy('./default_profile/**/*',
        '/storage/profiles/${user.username}/', { user: user })    }
    on add_user { user } => {
      exec add_user(user) on 'ad01.example.org'    }
```

BPMN representation algorithm of procedure calls from another procedure

The mechanism of calling a procedure from another procedure (or an anonymous procedure for reacting to an event) requires a separate discussion. Figure 2 presents the BPMN diagram for the module presented above. Calling a procedure from another procedure is asynchronous. Execution of the parent procedure (in the context of machine on which it is being executed) is stopped, the state of variables is remembered, and then the execution of called procedure is started (in the context of machine for which it was called, not necessarily the same for which the parent procedure is being executed). The diagram in figure 2 shows that in response to event named add_user, the add_user procedure will be invoked for the host ad01.example.org with the parameter user with the same value as was attached to event object add_user. Control between procedures is always done through the *Message Flow* elements. Due to simplification of the reverse translation algorithm (from BPMN to 'spa-lang'), the convention was applied that each procedure and handler have separate *Pool* objects (even if both pools are assigned to the same machine), which makes the reverse mapping process easier.

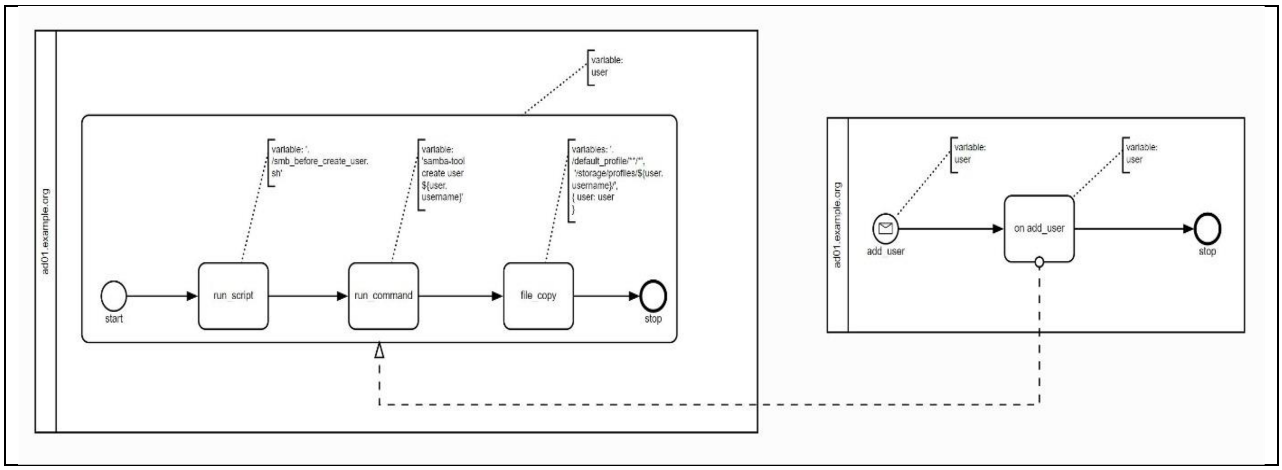

Figure 2. Presentation of module and procedure calls from 'spa-lang' to BPMN diagrams

Rules for mapping conditional statements to BPMN

Conditional statements are mapped in BPMN to *Gateway* objects (exclusive), whereby using the directly nested conditional expression *if... else if... else...* in the BPMN representation a single gate is generated with a greater number of outgoing *sequence flow* connections. The example given in figure 3 also shows how events are generated using the *emit* keyword in the 'spa-lang' code. The current version of the mapping algorithm adopts the convention that if the name of the emitted event ends with the suffix "error", the emitted event is additionally marked with an error icon. In the example in figure 3, it is also the last action of the procedure (the return directive) completes execution, so the system marked this event as final (in bold).

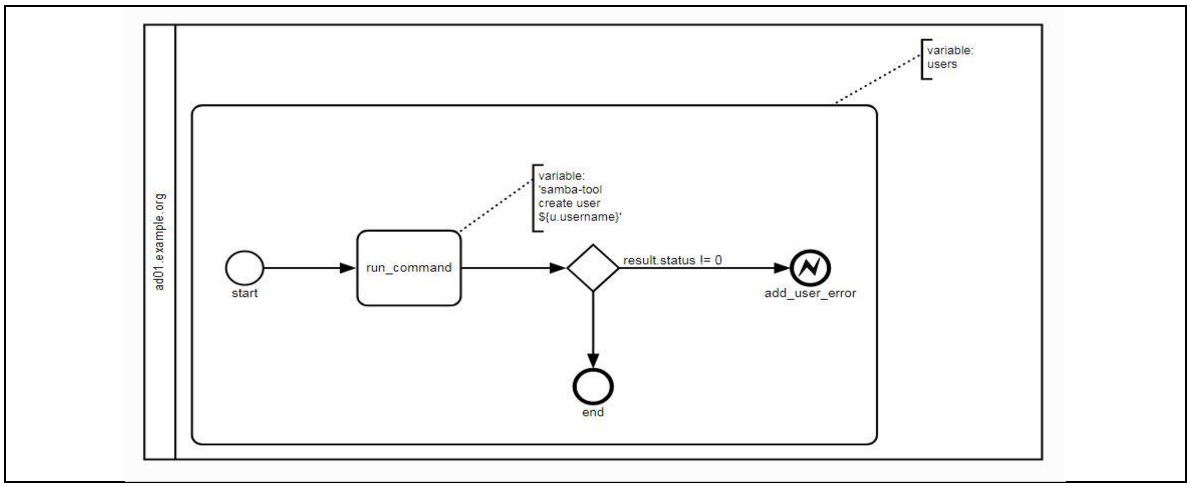

Figure 3. Presentation of condition from 'spa-lang' to BPMN diagrams

Loop to BPMN mapping algorithm

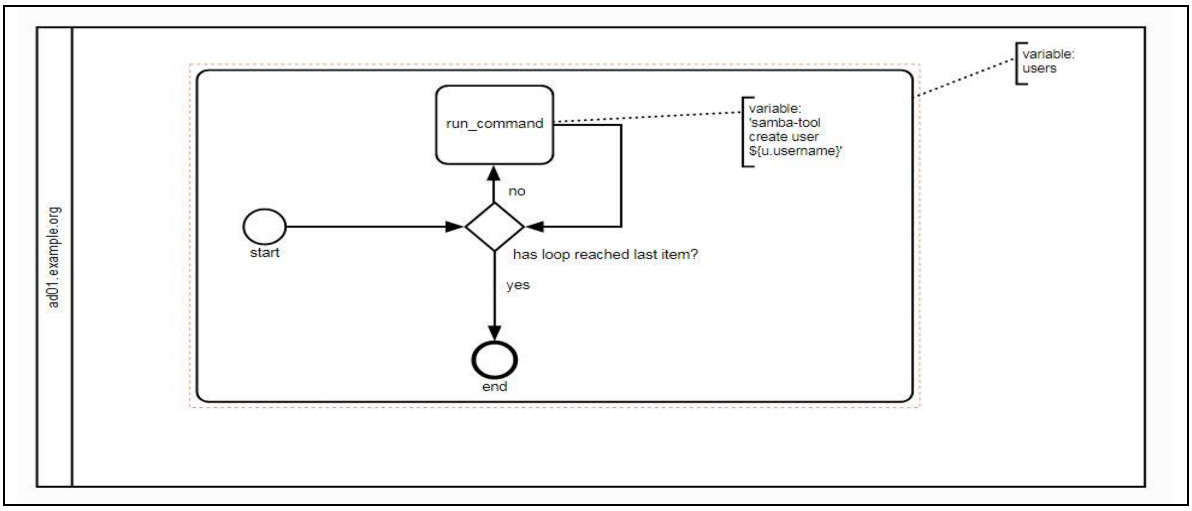

Figure 4. Mapping of condition from 'spa-lang' to BPMN diagrams

The BPMN standard does not have any specific representation signaling the execution of the loop, therefore the mapping according to the scheme shown in figure 4 was used here. The loop expression begins with an exclusive type gateway (*Gateway*), then the control flows sequentially into the loop body and returns to the gate. If the loop end condition is met, control flows sequentially to the next expression after loop (end of procedure, signaled by end event).

As part of the project, the BPMN editor web application (to the extent supported by 'spa-lang') was also developed. Since the mapping algorithms presented above use only specific configurations of BPMN elements, it was necessary to create a specialized editor supporting the construction of diagrams that can be also next translated inversely to 'spa-lang' syntax (Fig. 5).

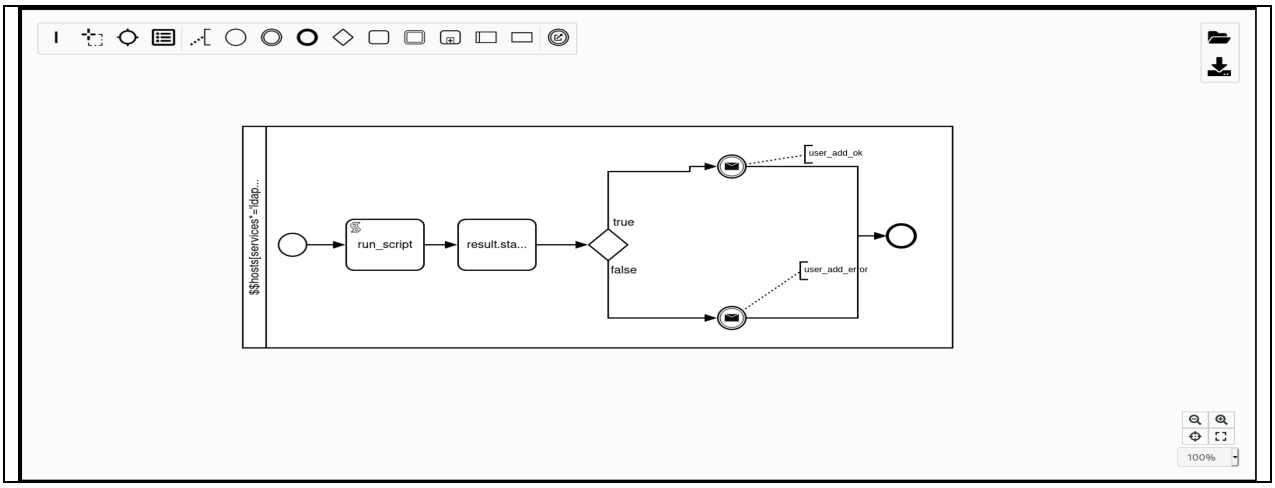

Figure 5. BPMN editor web application for 'spa-lang'

CONCLUSIONS

The research work described in this paper results in the resolution of the previously indicated technical uncertainties necessary for the implementation of key functionality of iSPA system. Thanks to the algorithm for translating the 'spa-lang' language into safety procedures, it is possible to carry out frequently performed safety-related activities directly by people who decide to change safety rules (e.g. management staff), without the need to involve technical staff each time and without the need for inspections and receipt of correctness of the performance of certain administrative activities. In the practice of many enterprises and institutions, such solution to security issues will translate into significant savings in costs related to security due to the enormous simplification and shortening of management processes.

REFERENCES

Argyropoulos, N., Mouratidis, H., and Fish, A. 2019 "Enhancing secure business process design with security process patterns." *Software & Systems Modeling*, pp. 1-23.

Chang, J. F. 2016 "Business process management systems: strategy and implementation." CRC Press.

van Dijk, R., Creeten, Ch., van der Ham, J., and van den Bos, J. 2017 "Model-driven software engineering in practice: privacy-enhanced filtering of network traffic." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ACM, pp. 860-865.

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. 2013 "Business process management." Springer.

Elsemary, H. 2016 "Mitigating Malware Attacks via Secure Routing in Intelligent Device-to-Device Communications." *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics*, Springer International Publishing, pp. 205-214.

Gates, Ch., Li, N., Xu, Z., Chari, N.S., Molloy, I., and Park. Y. 2014 "Detecting insider information theft using features from file access logs." *Proceedings of the European Symposium on Research in Computer Security*, Springer, pp. 383-400.

Goel, S., and Shawky, H. A. 2009. "Estimating the Market Impact of Security Breach Announcements on Firm Values," *Information & Management* (46:7), pp. 404-410.

Gross, M., and Ho, S. M. 2021. "Collective Learning for Increasing Cyber Defense Consciousness: An Activity System Analysis," *Journal of Information Systems Education* (32:1), pp. 65-76.

Ho, S. M., Hancock, J. T., Booth, C., and Liu, X. 2016. "Computer-Mediated Deception: Strategies Revealed by Language-Action Cues in Spontaneous Communication," *Journal of Management Information Systems* (33:2), pp. 393-420.

Ho, S. M., Ocasio-Velázquez, M., and Booth, C. 2017. "Trust or Consequences? Causal Effects of Perceived Risk and Subjective Norms on Cloud Technology Adoption," *Computers & Security* (70), pp. 581-595.

Ho, S. M., and Hancock, J. T. 2018. "Computer-Mediated Deception: Collective Language-Action Cues as Stigmergic Signals for Computational Intelligence," *Proceedings of the 2018 51th Hawaii International Conference on System Sciences (HICSS-51)*, Big Island, Hawaii: University of Hawaii, pp. 1671-1680.

Ho, S. M., and Gross, M. 2021. "Consciousness of Cyber Defense: A Collective Activity System for Developing Organizational Cyber Awareness," *Computers & Security* (108:102357), pp. 1-18.

Kim, B.H., Kim, K.C., Hong, S.E., and Oh, S.Y. 2017 "Development of cyber information security education and training system." Multimedia Tools and Applications (76:4) pp. 6051-6064.

Krym, T., Chomatek, L., and Poniszewska-Maranda, A. 2021. "Process business modelling of emerging security threats with BPMN extension." *Proceedings of AIS SIG SEC Pre-ICIS 2021 Workshop of Information Security and Privacy (WISP 2021)*, International Conference of Information Systems (ICIS 2021), Austin, USA, Chapter 8, Publisher: AIS.

Kumar, A. 2018 "Business Process Management." Taylor & Francis.

Maines, C. L., Llewellyn-Jones, D., Tang, S., and Zhou, B. 2015 "A cyber security ontology for BPMN-security extensions." *Proceedings of IEEE International Conference on*

Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, IEEE, pp. 1756-1763.

Maines, C. L., Zhou, B., Tang, S., and Shi, Q. 2016. "Adding a third dimension to BPMN as a means of representing cyber security requirements." *Proceedings of 9th International Conference on Developments in eSystems Engineering*, IEEE, pp. 105-110.

Nazir, A., Alam, M., Malik, S.U.R., Akhunzada, A., Nadeem Cheema, M., Khurram Khan, M., Ziang, Y., Khan, T., and Khan, A., 2017 "A high-level domain-specific language for SIEM (design, development and formal verification)." Cluster Computing (20) pp. 2423-2437.

Rodriguez, A., Fernandez-Medina, E., and Piattini, M. 2007. "A BPMN extension for the modeling of security requirements in business processes." *IEICE Transactions on information and systems*, (90:4), pp. 745-752.

Salnitri, M., Dalpiaz, F., and Giorgini, P. 2014 "Modeling and verifying security policies in business processes." *Enterprise, business-process and information systems modeling*, Springer, pp. 200-214.

Sharghi, H., and Kamran, S. 2017 "An expressive event-based language for representing user behavior patterns." *Journal of Intelligent Information System* (49) pp. 435-459.

Zaghetto C., Henrique, L., Aguiar, M., Zaghetto, A., Ralha, C.G., and de Barros Vidal, F. 2017 "Agent-based framework to individual tracking in unconstrained environments." *Expert Systems with Applications* (87) pp. 118-128.

Zareen, S., Akram, A., and Ahmad Khan, S. 2020 "Security Requirements Engineering Framework with BPMN 2.0. 2 Extension Model for Development of Information Systems." *Applied Sciences* (10:14), pp. 4981.