A collaborative Tool to Support Junior Developers in Secure Software Development

Maria Teresa Baldassarre[1], Vita Santa Barletta[1], Danilo Caivano[1], Giovanni Dimauro[1] and Antonio Piccinno[1]

[1] *University of Bari Aldo Moro, Via Orabona 4, 70125, Bari, Italy*

Abstract

In today's innovation and transformation scenarios, such as smart cities that transversally invest many technological fields and sectors of intervention, the security factor plays a primary role in software development projects. Currently, existing methodologies support developers only in the early stages of the software development life cycle (SDLC) and not in all of them. Therefore, it is necessary to provide operational guidelines in order to integrate privacy requirements and, consequently security, in software applications. Taking into account these needs, the research work presents a tool that supports team decisions to integrate privacy and security requirements in all software development phases regardless of the development methodology adopted.

Keywords

Privacy by Design, Human-Centered Privacy, Privacy Software Application

1. Introduction

Nowadays, organizations are undertaking increasingly complex projects in globalized, uncertain and dynamic environments [1]. Some of the factors that increase or generate project complexity are the emergence of new technologies [2], the growing sophistication of software project scopes characterized by challenging technical, time and cost requirements [3] and the increasing number of stakeholders involved [4]. Thus, to assure a successful software project [5], it is particularly important to stress concepts such as data and information security in all phases of the software development life cycle (SDLC) [6]. Software errors can be introduced by disconnections and miscommunications during planning, development, testing and maintenance of the components [7].

So, in order to satisfy security requirements, it becomes necessary to identify the key elements associated to the privacy and security-oriented software development, and to support decision making in all phases of the software life cycle.

Accordingly, this paper presents the key elements for privacy-oriented software development formalized in a knowledge base and a collaborative tool that integrates these elements. The tool supports decision making in all the software development lifecycle phases in order to integrate privacy and security requirements.

The paper is organized as follows. Section 2 discusses related works. Section 3 describes the tool and Section 4 reports the evaluation of collaborative tool and the obtained results. Finally, conclusions are given in Section 5.

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy
EMAIL: mariateresa.baldassarre@uniba.it (A. 1); vita.barletta@uniba.it (A. 2); danilo.caivano@uniba.it (A. 3); giovanni.dimauro@uniba.it (A. 4); antonio.piccinno@uniba.it (A. 5)
ORCID: 0000-0001-8589-2850 (A. 1); 0000-0002-0163-6786 (A. 2); 0000-0001-5719-7447 (A. 3); 0000-0002-4120-5876 (A. 4); 0000-0003-1561-7073 (A. 5)

2. Related Works

Software development requires integration of security and privacy to address threats related to cyber-attack [8]. So, for secure software development, it is particularly important to stress concepts such as data and information security in project activities which deal with or target aspects such as integrity, availability, and confidentiality [9,10]. This imposes, see for example the provisions on the subject dictated by the GDPR, "General Data Protection Regulation" [11], the adoption of processes and tools to mitigate the risks related to security and data privacy.

Privacy by Design [12] is a fundamental concept for software development as it requires data protection and the implementation of technical and organizational measures to protect the rights of users [13]. Confidentiality, integrity, availability, unlikability, transparency and intervenability are considered the six protection goals that provide a common scheme for addressing the legal, technical, economic and social dimension of privacy and data protection in complex IT systems [14].

The methodologies include two categories, *Security-based adaptations*, i.e., Privacy Engineering methodologies that are based and originated from security engineering methodologies, and *Privacy-Friendly systems* that aim to embed privacy into every step of the software development life cycle [15]. Therefore, a number of privacy requirements methodologies have been introduced in order to assist system designers and developers to analyze and elicit privacy requirements for different software systems and in different architectures [16].

LINDDUN [17] is a privacy threat modeling methodology that supports analysts in systematically eliciting and mitigating privacy threats in software architectures. The identified privacy threats are mapped with the existing privacy-enhancing technologies (PETs) [18]. PriS methods describes the effect of privacy requirements on business processes and facilitate the identification of the system architecture that best supports the privacy-related business processes [19]. SQUARE [20] security quality requirements engineering methodology supports the elicitation of privacy requirements at the early stages of software life cycle. It consists of nine steps which include what techniques will be used to elicit security requirements then categorize, prioritize, and inspect the requirements. STRAP [21] is a light-weight structured analysis of privacy vulnerabilities into the software development cycle. RBAC [22] method considers privacy requirements as constraints on permissions and user roles in order to define access control policies. PRIPARE introduces how privacy requirements should be incorporated into the software development life cycle [23].

Reviewing each of these methodologies, privacy requirements are addressed during the first phases of the SDLC, but never in all the phases. Therefore, it is necessary to integrate privacy in all phases of the development. Considering this need, the research work presents a tool to support the development team in all phases of the SDLC.

3. Collaborative Tool Proposal

Starting from the work illustrated in [24], the tool has been improved to overcome the limitations highlighted in the first experiment. It integrates the formalization of the privacy key elements in order to support decisions and choices in all phases of the software development (Figure 1). The five elements identified are Privacy by Design Principles, Privacy Design Strategies, Privacy Patterns, Vulnerabilities and Context.

Privacy by Design Principles [12] that describe the privacy model, each of which specify actions and responsibilities:

1. anticipate and prevent privacy-invasive events before they happen (*Proactive not Reactive*);
2. build privacy measures directly into any given ICT system or business practice by default (*Privacy as the Default*);
3. embed privacy into the design and architecture (*Privacy Embedded in to Design*);
4. accommodate all legitimate interests and objectives in a positive-sum manner, not through a zero-sum approach involving unnecessary trade-offs (*Full Functionality*);
5. ensure secure life-cycle management of information (*End-to-End Lifecycle Protection*);

6. keep component parts of systems and operations of business practices visible and transparent, to user and providers alike (*Visibility and Transparency*);
7. respect and protect interests of the individual, above all (*Respect for User Privacy*).

Privacy Design Strategies [25] that help IT architects to support Privacy by Design early in the software development life cycle. They are divided in two different categories:

- Data-Oriented Strategies focus on the privacy-friendly processing of the data (*Minimize, Hide, Separate, Abstract*).
- Process-Oriented Strategies focus on the process surrounding the responsible handling of personal data (*Inform, Control, Enforce, Demonstrate*).

Privacy Pattern [26] that underline the concept of pattern and can be used in all those situations where privacy is violated, and a user's personal data is no longer secure. Patterns are a literary format with which to capture the knowledge and experience of security experts, resulting in a structured document in the form of a template to which the security experts' knowledge is transferred [27,28].

Vulnerabilities within the code allow a malicious user to attack the application [7].

Context, that integrates *Architectural Requirements* to determine the flow of data within the system, components, roles and responsibilities; *Use Cases and Scenarios* to define all interactions within the system and protect the data from unauthorized reading and manipulation; and *Privacy Enhancing Technologies* (PETs) to protect the personal information handled by the applications.

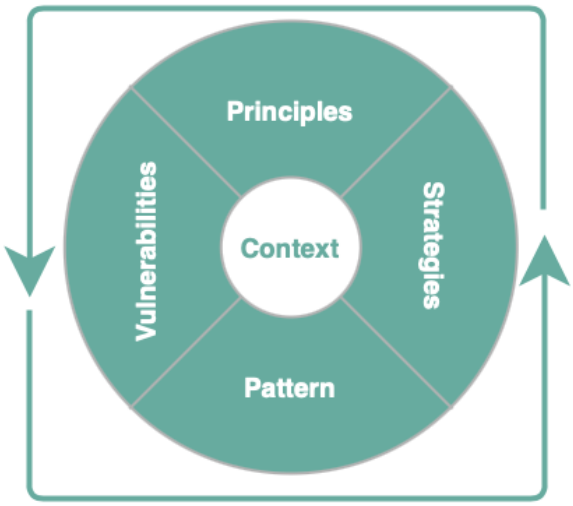

Figure 1: Privacy Knowledge Base: Key Elements

The collaborative tool supports the team's choices in all the software development phases in order to integrate security and privacy. At each stage of the SDLC, knowledge and skills are required to be able to develop a product that respects the principles of privacy and security by design. But many times, within the team certain knowledge are not present, such as knowing step by step the basic points of the GDPR [11] in order to provide a software product that is compliant. In order to overcome this limitation and provide operational guidelines to the development team, the tool allows you to select the necessary elements to be able to introduce privacy and security requirements throughout the software lifecycle. For example, starting from the vulnerabilities identified in the static code analysis the developer in addition to viewing a brief description of the vulnerability and an example, can also look at the privacy by design principles violated and view the privacy patterns that allow to mitigate the vulnerability and implement privacy strategies to correct the architectural flaws.

In Figure 2 are reported that OWASP Top 10 – 2017 [29], and the detail of Sensitive Data Exposure vulnerability. In addition, for each pattern identified for a particular vulnerability, it is shown how to implement it within a particular architecture chosen by the developer and on which he is working. If, for example, a privacy pattern or any other element of the knowledge base is not present, the team has the possibility to insert it in order to extend it. Figure 3 shows the addition of a pattern where it requires

the title, which privacy strategy it implements, the context, the problem, the solution, the UML diagram, the vulnerability it mitigates, and the architecture.

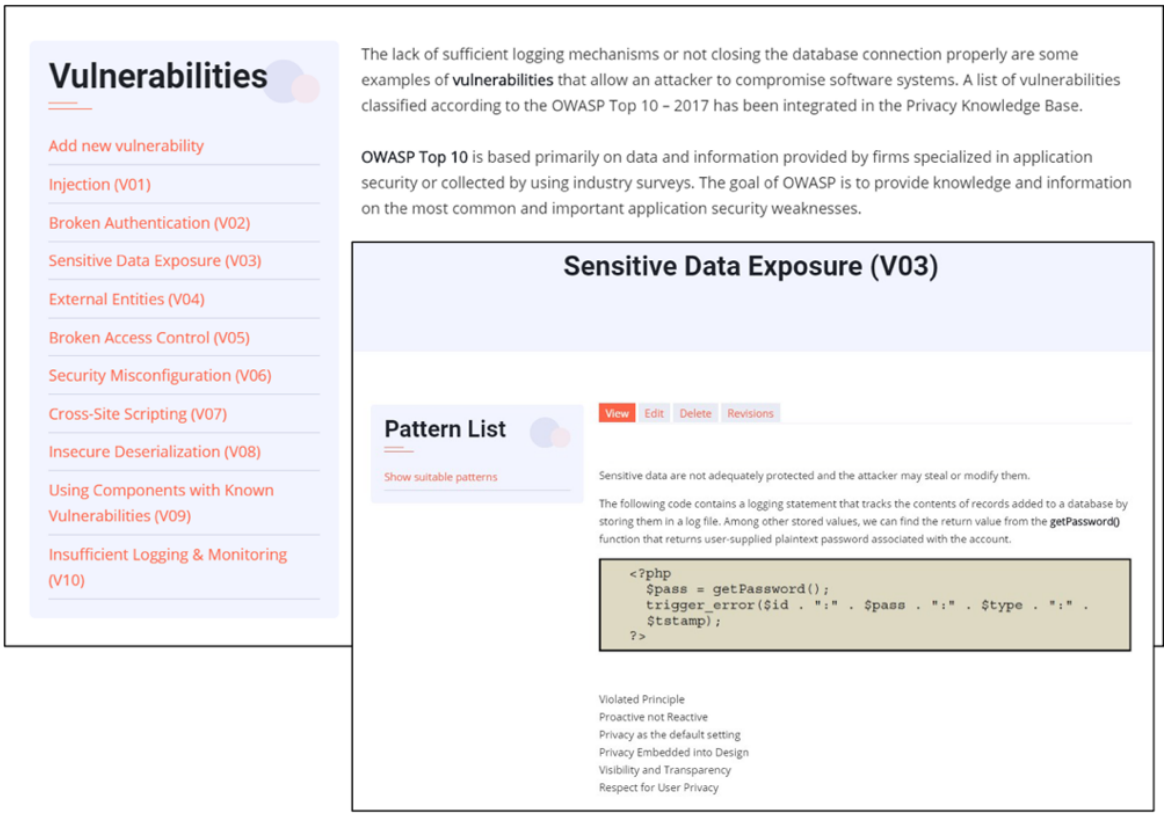

Figure 2: Vulnerabilities in Privacy Knowledge Base

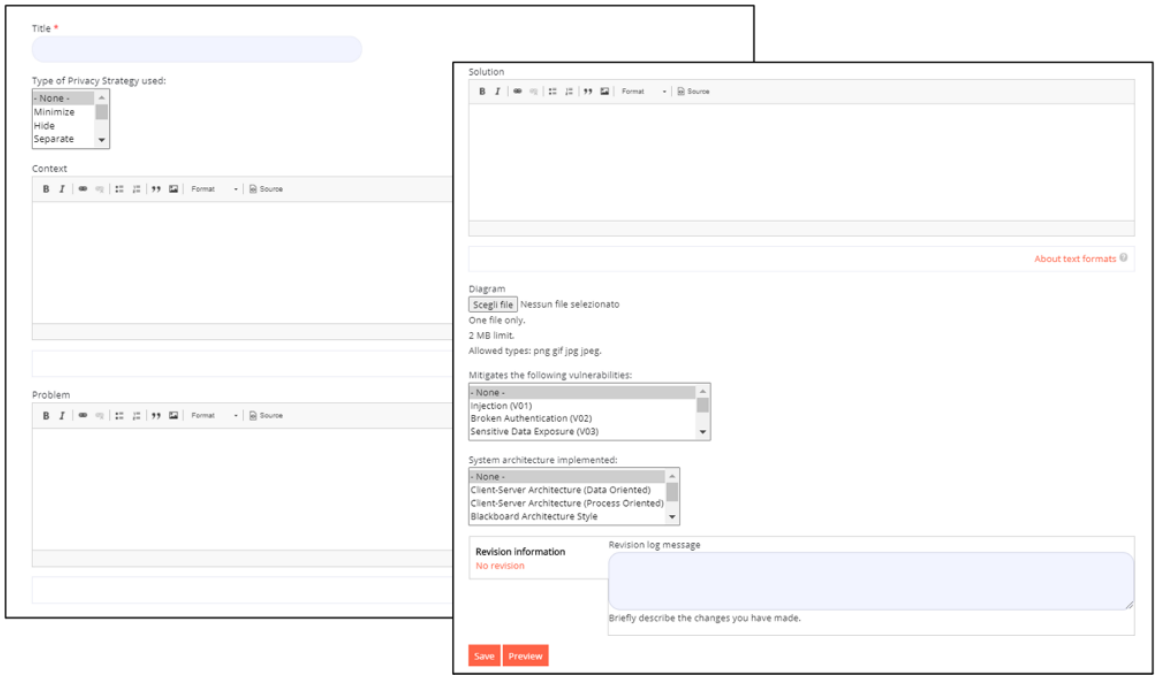

Figure 3: Privacy Patterns in Privacy Knowledge Base

4. Evaluation

To evaluate the potential benefits of the tool in order to integrate privacy and security in software, we conducted a pilot experiment with 8 bachelor students in computer science, software engineering curriculum.

The students have taken the traditional software engineering course and have no expertise in both software development and in applying privacy oriented methodologies. The average is 22 years old. They were divided into 2 groups and asked to re-engineer a system developed by their colleagues in previous years during the software engineering course.

The system called "Civic Sense" allows a citizen to report failures, problems, malfunctions and, in general, events relevant to an entity that provides services or manages infrastructure of public interest (electricity, roads and streets, urban safety, etc.).

As in the previous study in which we involved junior developers [24], the tasks assigned to the two groups are the following:

- *Task 1*: Identify the principles of Privacy by Design violated by vulnerability.
- *Task 2*: Identify the Privacy Design Strategies to be implemented in the system to respect the principles of Privacy by Design.
- *Task 3*: Identify the privacy patterns that substantiate the Privacy Design Strategies.
- *Task 4*: Identify the Data Strategies Component to implement in Target Architecture in order to re-engineer the system from a privacy point of view.
- *Task 5*: Identify the Processes Component Strategies to implement in Target Architecture in order to re-engineer the system from a privacy point of view.

In order to perform the required tasks, each group was provided with a security report containing the list of vulnerabilities identified during the static code analysis of the system (Issue ID, Category, Severity, Short Description). The following is an example of the *Privacy Violation* category.

Abstract: The file loginn.php mishandles confidential information on line 12, which can compromise user privacy and is often illegal.
Explanation: Privacy violations occur when:
1. Private user information enters the program.
2. The data is written to an external location, such as the console, file system, or network.
Example: The following code contains a logging statement that tracks the contents of records added to a database by storing them in a log file. Among other values that are stored is the return value from the getPassword() function that returns user-supplied plaintext password associated with the account.

```
<?php
$pass = getPassword();
trigger_error($id . ":" . $pass . ":" . $type . ":" . $tstamp); ?>
```

The code in the example above logs a plaintext password to the application eventlog. Although many developers trust the eventlog as a safe storage location for data, it should not be trusted implicitly, particularly when privacy is a concern.
Private data can enter a program in a variety of ways:
- Directly from the user in the form of a password or personal information - Accessed from a database or other data store by the application
- Indirectly from a partner or other third party
Sometimes data that is not labeled as private can have a privacy implication in a different context. For example, student identification numbers are usually not considered private because there is no explicit and publicly-available mapping to an individual student's personal information. However, if a school generates identification numbers based on student social security numbers, then the identification numbers should be considered private.
Security and privacy concerns often seem to compete with each other. From a security perspective, you should record all important operations so that any anomalous activity can later be identified. However, when private data is involved, this practice can create risk.

Then, starting from the security report, the students carried out each task in order. Each task was done in different sessions, one for each task, and each session was done in parallel. This was done to prevent the two groups from communicating with each other and knowing in advance the task to be performed. In addition, each group was observed by a researcher.

Keeping in mind that the research goal is to support the development team in all phases of the security and privacy oriented software lifecycle, we did not set out to investigate the developers' productivity. The focus was to observe the interaction with the tool to operationally implement guidelines in privacy-oriented software development and collect information to improve the tool.

Table 1 and Table 2 present the results obtained using the tool in re-engineering Civic Sense.

All tasks were done correctly although Task 4 of Group A was not accomplish by the time limit. In each session, it was critical to observe task execution to understand user behavior in being able to retrieve the information needed in reengineering the system.

The students' feedback was positive in being able to use a single tool that would allow them to view all the information needed to integrate privacy and security requirements operatively. Figure 4 shows an example of the architecture with the identification of privacy patterns that implement the privacy design strategies.

And this was confirmed by the SUS [30] [31] questionnaire conducted after at the end of the five sessions and the execution of the various tasks. The SUS score obtained was 89,9.

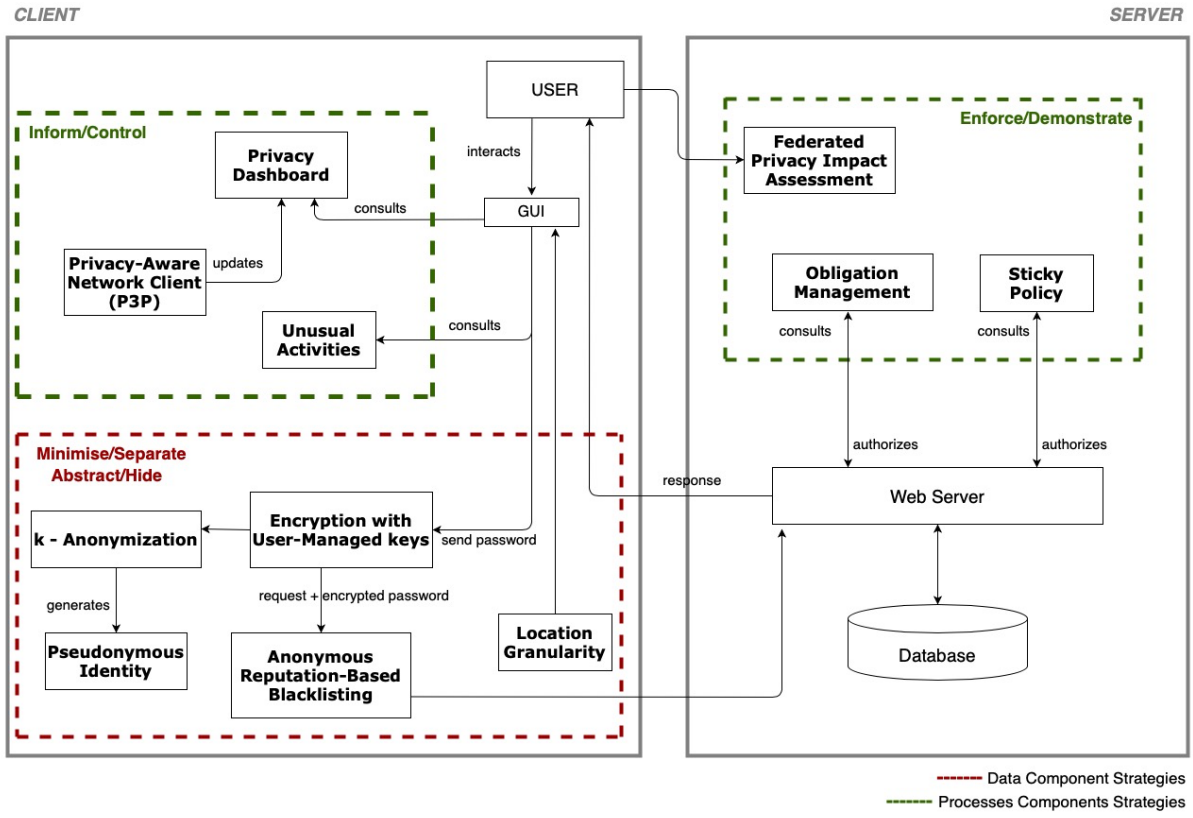

Figure 4: Target Architecture Result

Table 1
Task Results Group A

Task	Goal	Time limit	Time spent
Task 1	PbD violated	0h 30m	0h 21m
Task 2	Privacy Design Strategies to be implemented	0h 30m	0h 27m
Task 3	Privacy Patterns	1h 0m	0h 58m
Task 4	Data Component Strategies	1h 30m	**1h 33m**
Task 5	Process Component Strategies	1h 30m	1h 29m

Table 2
Task Results Group B

Task	Goal	Time limit	Time spent
Task 1	PbD violated	0h 30m	0h 29m
Task 2	Privacy Design Strategies to be implemented	0h 30m	0h 26m
Task 3	Privacy Patterns	1h 0m	0h 57m
Task 4	Data Component Strategies	1h 30m	1h 29m
Task 5	Process Component Strategies	1h 30m	1h 28m

5. Conclusions

In this research work, we analyzed the potential benefits of a collaborative tool that supports developers in order to integrate privacy and security in software development. The tool integrates the key elements identified in privacy-oriented software development (Principles of Privacy Design, Privacy Design Strategies, Privacy Patterns, Vulnerabilities, Context) and formalized in a knowledge base. The knowledge base shows that it is able to translate best practices for both secure application development and data privacy, into operational guidelines, software architectures and code structures to be used.

The tool allows to provide all the information needed to integrate privacy and security requirements during each stage of the re-engineering process. Students in the experimentation were positively supported despite not having specific skills in security by design and privacy by design. Indeed, the results demonstrate that the proposed tool supports developers in collaborative contexts integrating privacy and security in software development processes.

The positive results obtained by this study with bachelor students and the previous one with junior developers, encouraged us to plan for further case studies and evaluate developer's productivity with the collaborative tool.

6. References

[1] Barletta, V.S., Caivano, D., Dimauro, G., Nannavecchia, A., Scalera, M, (2020). Managing a Smart City Integrated Model through Smart Program Management. Appl. Sci. 2020, 10, 714. https://doi.org/10.3390/app10020714

[2] Pesare, E., Roselli, T., Rossano, V., Di Bitonto, P. (2015). Digitally enhanced assessment in virtual learning environments. Journal of Visual Languages & Computing, 31, 252-259. https://doi.org/10.1016/j.jvlc.2015.10.021

[3] Barletta, V.S.; Caivano, D.; Nannavecchia, A.; Scalera, M. A Kohonen SOM Architecture for Intrusion Detection on In-Vehicle Communication Networks. Appl. Sci. 2020, 10, 5062. https://doi.org/10.3390/app10155062
[4] Nonino F., Annarelli A., Gerosa S., Mosca P., Setti S., "Project Management: Driving Complexity", PMI® Italian Academic Workshop, (2018). ISBN 978-88- 9377-086-6
[5] Baldassarre M.T., Barletta V.S., Caivano D., Scalera M., (2019). Privacy Oriented Software Development. Communications in Computer and Information Science, 1010, pp. 18-32, https://doi.org/10.1007/978-3-030-29238-6_2
[6] Ellison, R. J., "Security and Project Management", 2006. Software Engineering Institute – Carnegie Mellon University
[7] Baldassarre, M.T., Barletta, V.S., Caivano, D., Scalera M., 2020. Integrating security and privacy in software development. *Software Qual J* (2020). https://doi.org/10.1007/s11219-020-09501-6
[8] Baldassarre, M. T., Barletta, V. S., Caivano, D., Raguseo, D., Scalera, M. (2019). Teaching Cyber Security: The Hack-Space Integrated Model. Proceedings of the *Third Italian Conference on Cyber Security*, Vol-2315, CEUR Workshop Proceedings.
[9] Jaccard, J. J., Nepal, S., "A survey of emerging threats in cybersecurity", Journal of computer and System Sciences, 2014. Volume 80, Issue 5, August 2014, pp. 973-993. https://doi.org/10.1016/j.jcss.2014.02.005
[10] Ardito, C.; Caivano, D.; Colizzi, L.; Dimauro, G.; Verardi, L. Design and Execution of Integrated Clinical Pathway: A Simplified Meta-Model and Associated Methodology. *Information* **2020**, *11*, 362. https://doi.org/10.3390/info11070362
[11] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC.
[12] Cavoukian, A. (2010). Privacy by Design: The 7 Foundational Principles.
[13] Cavoukian, A. (2012). Operationalizing Privacy by Design: A Guide to Implementing Strong Privacy Practices. pp. 1–72.
[14] M. Hansen, M. Jensen and M. Rost (2015). Protection Goals for Privacy Engineering, IEEE Security and Privacy Workshops, San Jose, CA, 2015, pp. 159-166.
[15] Al-Slais, Y., 2020. Privacy Engineering Methodologies: A survey. International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Sakheer, Bahrain, 2020, pp. 1-6, doi: 10.1109/3ICT51146.2020.9311949.
[16] Pattakou, A., Mavroeidi, A., Diamantopoulou, V., Kalloniatis, C., Gritzalis, S., 2018. Towards the Design of Usable Privacy by Design Methodologies. IEEE 5th International Workshop on Evolving Security & Privacy Requirements Engineering (ESPRE), Banff, AB, Canada, 2018, pp. 1-8, doi: 10.1109/ESPRE.2018.00007.
[17] Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W., 2011. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, vol. 16, no. 1, pp. 3-32, 2011.
[18] Van Blarkom, G.W., Borking, J.J., and Olk, J.G.E. (2003). Handbook of Privacy and Privacy-Enhancing Technologies. The Case of Intelligent Software Agents. College Bescherming Bersoonsgegevens, ISBN 90-74087-33-7
[19] Kallpniatis, C., Kavakli, E., Gritzalis, S. (2008). Addressing privacy requirements in system design: the PriS method. *Requirements Engineering*, vol. 13, pp 241-255. Springer-Verlag. https://doi.org/10.1007/s00766-008-0067-3
[20] Mead, N.R., Stehney, T., 2005. Security quality requirements engineering (SQUARE) methodology, *ACM*, vol. 30, no. 4, pp. 1-7, 2005
[21] Jensen, C., Tullio, J., Potts, C., Mynatt, E.D., 2005. STRAP: a structured analysis framework for privacy, *Georgia Institute of Technology*, 2005.
[22] He, Q., Anton, A.I., 2003. A framework for modelling privacy requirements in role engineering, *In: Proceedings of REFSQ*, vol. 3, pp. 137-146, 2003.
[23] Notario, N., Crespo, A., Martìn, Y. S., Del Alamo, J. M., Le Métayer, D., Antignac, T., Kung, A., et al. (2015). PRIPARE: Integrating Privacy Best Practices into a Privacy Engineering Methodology. *IEEE Security and Privacy Workshops*, San Jose, CA, pp. 151-158. https://doi.org/10.1109/SPW.2015.22

[24] Baldassarre, M.T., Barletta, V.S., Caivano, D., Piccinno, A., 2020. A Visual Tool for Supporting Decision-Making in Privacy Oriented Software Development. In Proceedings of the International Conference on Advanced Visual Interfaces (AVI '20). Association for Computing Machinery, New York, NY, USA, Article 45, 1–5. https://doi.org/10.1145/3399715.3399818
[25] Hoepman, J.-H. (2014). Privacy Design Strategies. In IFIP, ICT Systems Security and Privacy Protection (pp 446-459). Springer Berlin Heidelberg.
[26] Privacy Patterns, https://privacypatterns.org. Resource document. UC Berkeley, School of Information. Accessed 10 January 2020.
[27] M. Schumacher "B. Example Security Patterns and Annotations" in Security Engineering with Patterns, 2003, pp. 171-178.
[28] Moral-García, S., Ortiz, R., Moral-Rubio, S., Vela, B., Garzás, J., & Fernández-Medina, E. (2010). A new pattern template to support the design of security architectures.
[29] The Open Web Application Security Project, 2019. OWASP Top 10–2017. The Ten Most Critical Web Application Security Risks. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project.
[30] Brooke, J. “SUS - A quick and dirty usability scale.”, in Usability evaluation in industry, CRC Press, 1996.
[31] Borsci, S., Federici, S., & Lauriola, M. (2009). On the dimensionality of the System Usability Scale: a test of alternative measurement models. Cognitive Processing, 10(3), 193-197. doi: 10.1007/s10339-009-0268-9