2021

A Tool for Improving Privacy in Software Development

Maria Teresa Baldassarre
university of bari, italy, mariateresa.baldassarre@uniba.it

Vita Santa Barletta
university of bari, italy, vita.barletta@uniba.it

Danilo Caivano
university of bari, italy, danilo.caivano@uniba.it

Giovanni Dimauro
university of bari, italy, giovanni.dimauro@uniba.it

Antonio Piccinno
university of bari, italy, antonio.piccinno@uniba.it

A Tool for Improving Privacy in Software Development

Maria Teresa Baldassarre[1][0000-0001-8589-2850], Vita Santa Barletta[1][0000-0002-0163-6786], Danilo Caivano[1][0000-0001-5719-7447], Giovanni Dimauro[1][0000-0002-4120-5876] and Antonio Piccinno[1][0000-0003-1561-7073]

[1]Department of Computer Science,
University of Bari Aldo Moro, via Orabona 4, 70125 Bari, Italy
{mariateresa.baldassarre,vita.barletta,danilo.caivano,
giovanni.dimauro, antonio.piccinno}@uniba.it

Abstract. Privacy is considered a necessary requirement for software development. It is necessary to understand how certain software vulnerabilities can create problems for organizations and individuals. In this context, privacy-oriented software development plays a primary role to reduce some problems that can arise simply from individuals' interactions software applications, even when the data being processed is not directly linked to identifiable. The loss of confidentiality, integrity, or availability at some point in the data processing, such as data theft by external attackers or the unauthorized access or use of data by employees., represent some types of cybersecurity-related privacy events. Therefore, this research work discusses the formalization of 5 key privacy elements (Privacy by Design Principles, Privacy Design Strategies, Privacy Pattern, Vulnerabilities and Context) in software development and presents a privacy tool that support developers' decisions to integrate privacy and security requirements in all software development phases.

Keywords: Privacy by Design, Security by Design, Software Security Engineering.

1 Introduction

The development of software projects requires both tools and techniques for efficient management and the integration of privacy and security requirements. Like a cancer, vulnerable software can be invaded and modified to cause damage to previously healthy software, and infected software can replicate itself and be carried across networks to cause damage in other systems [1]. Several factors contribute to the increase of cyber-attacks such as the emergence of new technologies, the growing sophistication of software projects' scope characterized by challenging technical, time and cost requirements and the increasing number of stakeholders involved [2].

Therefore, software development requires the adoption of processes, tools and techniques that integrate security and privacy into all phases of the software lifecycle. The security is not simply embedded in the network security technologies in a software system through various types of security activities [3]. It is particularly important to stress

concepts such as confidentiality (the data is not made available or disclosed to unauthorized individuals, entities, or processes), integrity (the ability to make sure that a system and its data has not been altered or compromised) and availability (data or system must be available to authorized users at all times) in all phases of software development [4]. Privacy and security issues can be introduced by disconnects and miscommunications during the planning development, testing and maintenance of the components [5]. So, it is necessary to identify the key elements associated to the privacy and security oriented software development and provide the tools necessary to support developers' decisions all phases of the software life cycle [6].

The research work identifies developers as end users and formalizing key elements into a knowledge base allows for support on how and what to incorporate in order to develop security and privacy oriented software.

The paper is organized as follow. Section 2 discusses related works. Section 3 presents the privacy elements and Section 4 the Privacy Tool. Section 5 describes the methodologies. Section 6 reports the obtained results and, finally discussions and conclusions are given respectively in Section 7 and Section 8.

2 Related Works

Software is considered secure, that is a prerequisite to integrate privacy requirements, when it exhibits three interrelated properties [1]: the software executes correctly and predictably, even when confronted with malicious or anomalous inputs or stimuli (*Dependability*); the software itself contains no malicious logic or any flaws or anomalies that could be exploited or targeted as vulnerabilities by attackers (*Trustworthiness*); when the software is able to resist most attempted attacks, tolerate the majority of those it cannot resist, and recover with minimal damage from the very few attacks that succeed (*Resilience*). This imposes the adoption of processes, techniques and tools to mitigate the risks related security and data privacy [7].

Awanthika at al. [8] conducted a study on the difficulty of developing privacy preserving software systems. They identified 5 issues that developers faced when embedding privacy into the designs: Requirements in the design contradict with privacy requirement (*Contradiction*); Relating requirements to practice (*How can I implement the fair information practices?, embedding privacy is too complex*); Assurance (*I tried to use the theories in the design but I'm not sure if I really did, I think I did, I don't know if I have done it right*); Personal opinion (*storing raw data in the db does not affect the privacy*); Lack of knowledge (*I did not use the this privacy theory because I don't know them, I haven't heard of PIA*). Issues that the various identified methodologies would need to address in order to support developers.

Notario et al. [9] defined how privacy requirements should be incorporated into the software development life cycle; Deng et al. [10] identified 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) [11]. He & Anton [12] considered privacy requirements as constraints on permissions and user roles in order to define access control policies; Kallpniatis et al. [13] described the effect of privacy requirements on business processes and facilitate the identification of the system architecture that best

supports the privacy-related business processes. Security quality requirements engineering methodology supports the elicitation of privacy requirements at the early stages of software life cycle in the SQUARE methodology developed by Mead et al. [14]. It consists of nine steps which include what techniques will be used to elicit security requirements then categorize, prioritize, and inspect the requirements.

In each of these methodologies, privacy and security requirements are addressed during the first phase of the software development. Moreover, it is necessary to take into account that some elements as highlighted in [8] are not clearly outlined how to integrate them in the different contexts. So, starting from these needs, the research work aims to identify the key elements in supporting developers in the development of security and privacy-oriented software and to develop a tool to support developers at each stage of development.

3 Privacy Elements

The elements identified to support developers in all phases of the software development are Privacy by Design (PbD) Principles, Privacy Design Strategies, Privacy Pattern, Vulnerabilities and Context. These elements are formalized in a knowledge base that is the basis to support developer decisions in privacy-oriented software development.

Privacy by Design Principles [15]. These principles provide an excellent framework for system design, especially when privacy-sensitive data is involved [16]: PbD anticipates and prevents privacy threats (*Proactive not Reactive*); the privacy built into the system should not require any further user setup (*Privacy as the default setting*); privacy should not be implemented in response to a given event, but embedded in the design, IT system architecture and business logic (*Privacy Embedded into Design*); PbD needs both, privacy and security (*Full Functionality*); PbD ensures secure lifecycle management of information, end-to-end (*End-to-End Security*); component parts and operations remain visible and transparent to users and providers (*Visibility and Transparency*); architects and operators must consider the interests of the individual (*Respect for User Privacy*).

Privacy Design Strategies [17]. They translate vague legal norms in concrete design requirements. There are eight different privacy design strategies, divided over two different categories: data-oriented strategies and process-oriented strategies.
Data Oriented Strategies, called “Privacy-by-Architecture”, are methodologies to prevent attacks: limit the processing of personal data as much as possible (*Minimize*); protect personal data or make it unlikable or unobservable (*Hide*); separate the processing of personal data as much as possible (*Separate*); limit the detail in which personal data is processed (*Abstract*).
Process Oriented Strategies, also known as “Privacy-by-Policy”, aim to inform the user and to make decisions according to their needs: inform data subjects about the processing of their personal data (*Inform*); provide data subjects an adequate control over the processing of their personal data (*Control*); commit to processing personal data in a privacy-friendly way (*Enforce*); demonstrate personal data is being processed in a privacy-friendly manner (*Demonstrate*).

Privacy Pattern [18]. They 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. They standardize language in the context of privacy protection document common solutions to privacy problems; help designers identify and address privacy concerns; integrate privacy and security mechanisms. Each privacy pattern represents a solution on which privacy design strategies to implement and which vulnerabilities it mitigates.

Vulnerabilities. Vulnerabilities within the code allow a malicious user to attack the application. In this step of the research, the vulnerabilities of the OWASP Top Ten and those related to the GDPR (General Data Protection Regulation) category have been integrated [23]. An example of privacy vulnerability is Access Violation. Figure 1 shows code that supplies a user-controlled indicator to a method that fetches employee data. If an attacker supplies whitespace for parameter **p_xfeld**, no authorization checks are performed before returning an employee's personal and contact information.

Context. It is a key element for the system development and reengineering. The context is formalized of *Architectural Requirements* that determine the flow of data within the system, roles and responsibilities; *Use Cases and Scenarios* define all interactions with the system in order to protect the information from unauthorized reading and manipulation; *Privacy Enhancing Technologies* (PETs), tool and technologies that help protect the personal information handled by the applications.

```
    ...
        PARAMETERS: p_xfeld TYPE xfeld.
    ...
CALL FUNCTION 'BAPI_EMPLOYEE_GETDATA'
  EXPORTING
    employee_id      = emp_id
    authority_check  = p_xfeld
  IMPORTING
    return           = ret
  TABLES
    org_assignment   = org_data
    personal_data    = pers_data
    internal_control = con_data
    communication    = comm_data
    archivelink      = arlink.
    ...
```

Fig. 1. Access Control: Authorization Bypass.

4 Privacy Tool

A privacy tool has been implemented to support the team's choices in all the software development phases and to evaluate the research proposal. It implements the key elements identified as shown in Figure 2. Starting from a specific vulnerability of the OWASP Top 10, the tool shows a short description of the macro categories of vulnerabilities, an example that causes it and outlines how to mitigate it. For each

vulnerability, the privacy patterns that help mitigate it are shown. For example, Figure 3 shows the privacy patterns that implement the data-oriented strategy (*Hide*) for a client server architecture and to mitigate an injection vulnerability (*V01 – Injection*). Moreover, there is the possibility of exporting examples of identified patterns (Figure 4) as well as viewing the description and relationships with other elements.

Each pattern is structured as follows: *Name* that represents the problem addressed; *Context* that contains a generic description of the setting and specifies the conditions under which the privacy pattern should be applied; *Problem*, the situation requiring the application of privacy requirements; *Solution* describes the possibility of implementing certain mechanisms to solve the problem under investigation; *Diagram* represents the behavior and the structure of the pattern.

In this way, a developer with no specific knowledge and competences on privacy, can understand which principles of PbD are violated by the selected vulnerability and which privacy design strategies must be implemented through the use of privacy patterns. The relationship between these elements will support the developer in designing a Privacy Software Architecture. In addition, the tool integrates the result of static code analysis, derived from third party static code analysis tools, so that each vulnerability identified in the legacy system is associated to a privacy pattern, allowing to fix the architectural defects.

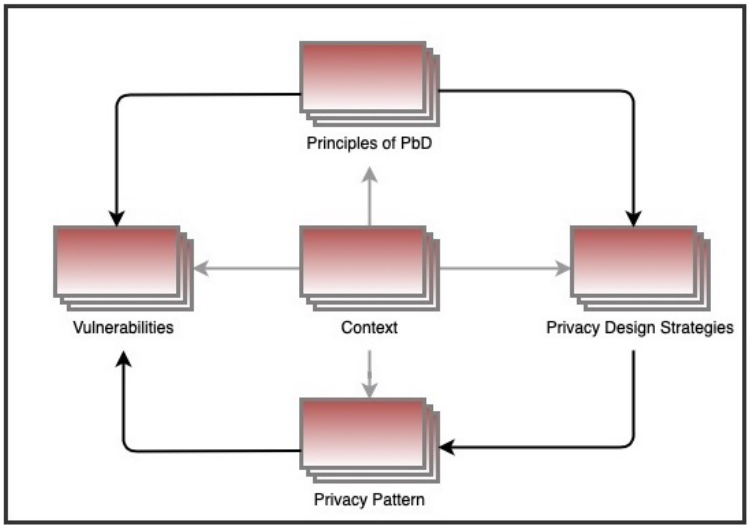

Fig. 2. Relationship between 5 elements.

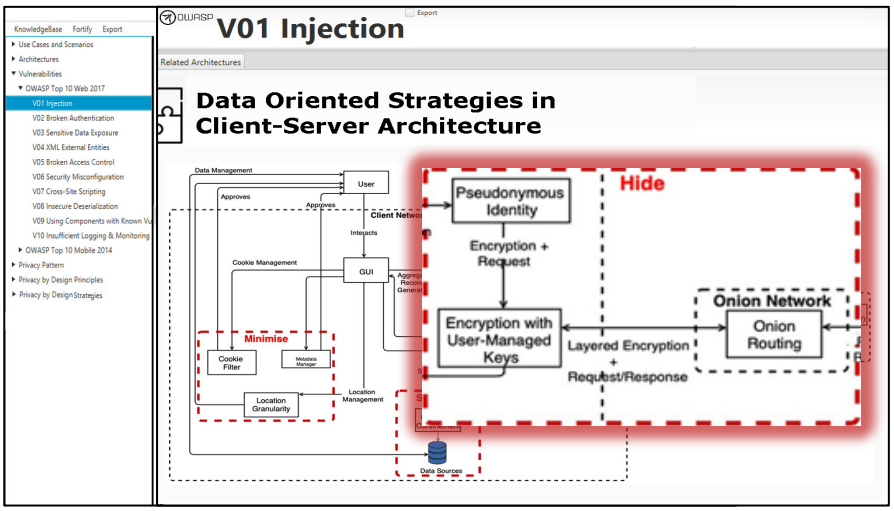

Fig. 3. Relationship between 5 elements

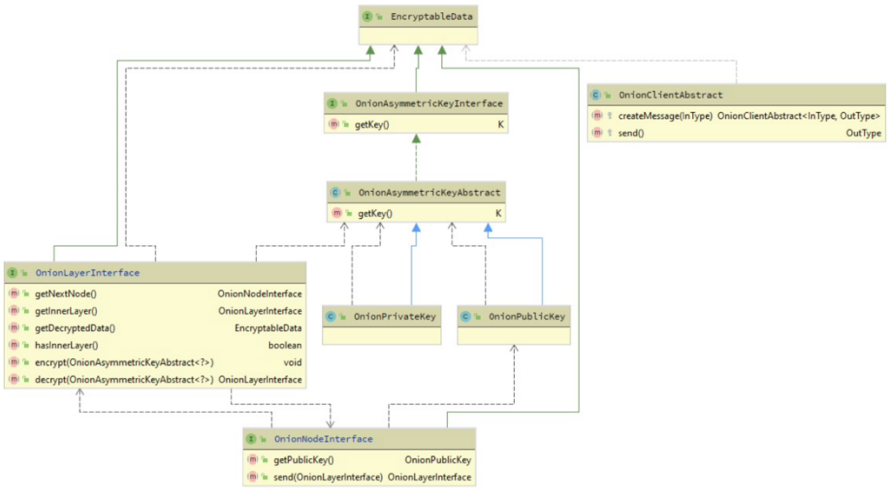

Fig. 4. Pattern Onion Routing

5 Methodology

The research leaded to define and implement a tool that integrates privacy and security elements for secure software development. Therefore, the main goal can be defined as follows: "*Analyze the potential utility of the tool with the purpose of providing operational guidelines from the point of view of developers in the context of privacy and security oriented software development*." The research carried out has been organized according to four steps (Figure 5): Definition, Planning, Execution, Analysis and Interpretation.

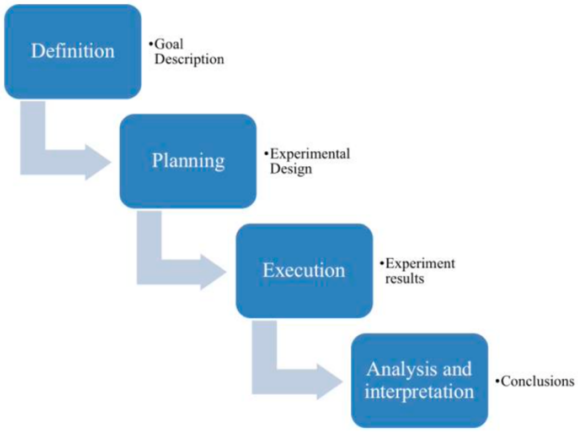

Fig. 5. Methodology.

Definitions. In this step we identified the research goal mentioned above and further refined it in terms of the following research questions [20]:

- RQ1: Do the key elements implemented in the tool allow to support the decision making in privacy and security software development?
- RQ2: Does the tool improve improves developers' choices in integrating privacy and security requirements?

Planning. The goal of this step was to identify the experimental variables, define the experiment design, select the subjects involved in the study and prepare the experimental materials needed for supporting the execution phase. More precisely:

- **Experimental Variables Identification**. The relevant variables that characterize the phenomena under study need to be identified. In the context of this research work they were identified through a literature review aimed to finding out the key elements that influence the privacy-oriented software development on one hand, and methodologies to support developers. The following are the 5 key elements identified:
  - *The 7 principles of Privacy by Design;*
  - *Privacy Design Strategies:* Data-oriented Strategies and Process-oriented Strategies;
  - *Privacy Pattern;*
  - *Vulnerabilities;*
  - *Context*: Use cases and Scenarios, Architectural Requirements and Privacy Enhancing Technologies.
- **Experiment Design**. The pilot experiment was identified to evaluate the potential utility of the privacy tool for supporting the team's choices in privacy-oriented software development. A total of 12 junior developers, representing our potential users in this research work. They were divided into three groups, 4 for each group (Group A, Group B and Group C), and asked to identify privacy and

security requirements in a legacy system to be reengineered. The legacy system was used by a public company for processing the personal data.
The evaluation took place in four parallel sessions and in different research laboratories, during which the tool was introduced by a member of the groups for 10 minutes. Thereafter, each group was asked to perform the following tasks:

- *Task 1.* Identify the principles of Privacy by Design violated by vulnerability in the legacy system.
- *Task 2.* Identify the Privacy Design Strategies to be implemented in the legacy 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.

- **Selection of the experimental sample**. For the selection of the experimental sample "Convenience sampling" was used, a specific type of non-probability sampling method based on data collected from population members [21]. The subjects were selected mainly because they were easiest to recruit for the study in comparison to testing the entire population that would have been too large and therefore impossible to include everyone. In the study 12 junior developers were involved with no specific knowledge and competences on software security and privacy. The reason for not involving security experts in this step of the research is to evaluate the potential utility of the tool in integrating privacy and security requirements into software development. The developers have a bachelor's degree in computer science and are employees in small companies. The average is 25 years old, and they have knowledge of Java and PHP programming.

Execution. The experiment was carried out as follows: the security report, containing the list of security and privacy vulnerabilities identified during the static code analysis of the legacy system (Issue ID, Category, Severity, Short Description), was given to each group. Then, starting from the security report, the developers carried out each task in order. Once completed each task, developers were requested to complete a SUS [22] [23] questionnaire through a Google form to measure the tool's usability. After having completed all tasks and questionnaire, the researchers facilitated a discussion focused on the privacy key elements that support the privacy-oriented software development.

Analysis and Interpretation. The goal of this research work is to observe the interaction with the tool to operationally implement privacy requirements in the software development. guideline in privacy-oriented software. Therefore, the correct identification of the privacy elements required in each task has been considered; the developer's assessment of the use of the tool to identify the elements needed.

6 Results

The results obtained using the privacy tool are described below. Task 1, 2 3 were successfully completed by both groups in the given time limit (we estimated 30 minutes for Task1 and Task 2; 60 minutes for Task 3; 90 minutes for the remaining two, Task 4 and Task 5).

Whereas, for task 4 and 5, the results of the three groups were different:

- Task 4 was completed by Group B, while Group A and Group B failed to complete it.
- Task 5 was completed by Group A and Group B, but Group C failed to complete it couldn't complete it in time.

Despite not having completed all the tasks, this represents a good result: development teams were able to correctly identify the principle of privacy by design violated by the vulnerabilities identified during static code analysis; the privacy strategies to implement to respect the Privacy by Design; the Privacy Design Strategies to implement; and somehow relate privacy patterns in the two Privacy Design Components, Data and Processes.

Table 1. Task Results Group A

Task	Success	Failure	Time spent
Task 1 - PbD violated	x	-	0h 17m
Task 2 - Privacy Design Strategies to be implemented	x	-	0h 25m
Task 3 - Privacy Patterns	x	-	1h 01m
Task 4 - Data Component Strategies	-	x	1h 35m
Task 5 - Process Component Strategies	x	-	1h 23m

Table 2. Task Results Group B

Task	Success	Failure	Time spent
Task 1 - PbD violated	x	-	0h 22m
Task 2 - Privacy Design Strategies to be implemented	x	-	0h 26m
Task 3 - Privacy Patterns	x	-	0h 59m
Task 4 - Data Component Strategies	x	-	1h 25m
Task 5 - Process Component Strategies	x	-	1h 27m

Table 3. Task Results Group C

Task	Success	Failure	Time spent
Task 1 - PbD violated	x	-	0h 25m
Task 2 - Privacy Design Strategies to be implemented	x	-	0h 30m
Task 3 - Privacy Patterns	x	-	0h 59m
Task 4 - Data Component Strategies	-	x	1h 49m
Task 5 - Process Component Strategies	-	x	1h 45m

For what concerns the qualitative analysis, we investigated how junior developers (our end users) considered the Privacy tool. The SUS score obtained was 89,7. We find these results to be encouraging, given that developers had little time to become familiar with the tool and they have no experience with privacy by design, privacy pattern and privacy design strategies.

During the discussion some suggestions have emerged: more detail on the relationships between vulnerabilities and privacy patterns; the possibility to export privacy patterns in different programming languages; the ability to identify privacy patterns needed when new use cases and scenarios arise.

7 Discussions

In this research work, the potential utility of privacy key elements through the use of the Privacy tool was evaluated. The study involved 12 junior developers and the results obtained are encouraging for supporting decision-making in privacy-oriented software development. One aspect that emerged during the discussion with the developers is to introduce new elements and relationships based on new cyber threats and changes in regulations such as GDPR.

An interesting point as it allows to enrich the underlying identified by the formalization of the 5 key privacy elements, improving privacy in software development. In this case, artificial intelligence (A.I.) plays a fundamental role. In particular, the privacy tool supports decision making in both new system development (forward mode) and re-engineering (backward mode). Therefore, the proper integration of privacy elements could be tested through static code analysis and penetration testing. This allows to validate the security of the system and consequently the privacy compliance, and to obtain a dataset that has been generated from all the formalized rules between the existing privacy elements. By training the A.I. model over the privacy existing solutions (dataset), it is possible to support team's decisions when, for example, new use cases and scenarios arise in the software development. Thus, the new privacy solutions obtained can be provided to the A.I. model to improve at various phases of software development the integration of privacy requirements and consequently support each decision of the development team. Figure 6 describes this idea in detail.

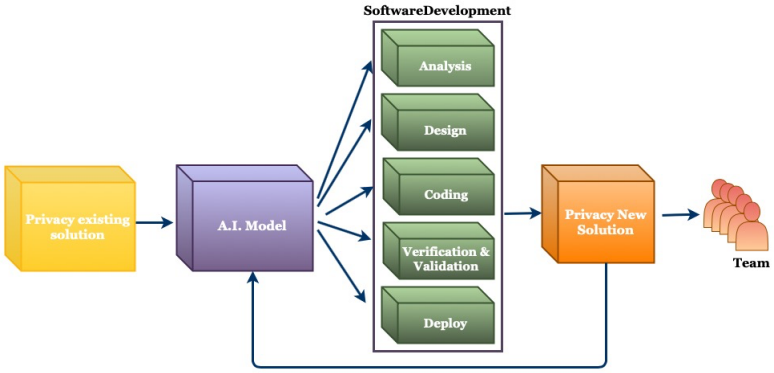

Fig. 6. A.I. model to support Privacy Oriented Software Development

8 Conclusions

This paper analyzed the key elements to integrate privacy, and consequently security, requirements in software development and to support decision-making in all phases of the software lifecycle. Privacy engineering require knowledge and skills, therefore the privacy tool implemented 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. It supports developers to fix vulnerabilities, and in the privacy-oriented reengineering of the legacy system [24], in assuming design decisions even though they may have no specific skills and knowledge about security and privacy. To evaluate the potential utility of the tool we conducted a pilot study with junior developers and the results suggest that the tool can meet the developers' needs in the context of privacy and security.

Given the results obtained, as future work we have planned to evaluate the developer's productivity adopting the tool; the development of artificial intelligence model in order to improve the decision making in all phases of the software development. Like to threat intelligence that can be used to inform decisions regarding the subject's response to the menace or hazard (Gartner), in this research privacy intelligence can be to better understand process threat data and their vulnerabilities, respond faster to implement privacy and security requirements, identify in the code analyzed a new vulnerability. The idea is to use machine learning to automate data collection and processing, take in unstructured data from different sources, and then connect the knowledge acquired by providing context on privacy and security requirements, and tactics, techniques, and procedures of privacy-oriented software development.

References

1. Goertzel, K., (2008). Enhancing the development life cycle to Produce Secure Software
2. 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
3. Masahito, S., Atsuo, H., Nobukazu, Y., Takanori, K., Hironori, W., Haruhiko, K., Takao, O., (2015). A Case-based Management System for Secure Software Development Using Software Security Knowledge. Procedia Computer Science, Volume 60, 2015, pp. 1092-1100, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2015.08.155.
4. Ellison, R. J., "Security and Project Management", 2006. Software Engineering Institute – Carnegie Mellon University
5. 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
6. Baldassarre, M.T., Barletta, V.S., Caivano, D., Piccinno, A. 2021. Integrating Security and Privacy in HCD-Scrum. CHItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter. Association for Computing Machinery, New York, NY, USA, Article 37, 1–5. https://doi.org/10.1145/3464385.3464746
7. 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.

8. Awanthika,S., Nalin A. G. A., (2018). Why developers cannot embed privacy into software systems? An empirical investigation. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (EASE'18). Association for Computing Machinery, New York, NY, USA, 211–216. https://doi.org/10.1145/3210459.3210484
9. 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
10. 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.
11. 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
12. 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.
13. 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
14. Mead, N. R., Stehney, T., 2005. Security quality requirements engineering (SQUARE) methodology, *ACM*, vol. 30, no. 4, pp. 1-7, 2005
15. Cavoukian, A. (2010). Privacy by Design: The 7 Foundational Principles.
16. Aad, I., & Niemi, V. (2010). NRC Data Collection and the Privacy by Design Principles.
17. Hoepman, J.-H. (2014). Privacy Design Strategies. In IFIP, ICT Systems Security and Privacy Protection (pp 446-459). Springer Berlin Heidelberg
18. Privacy Patterns, https://privacypatterns.org. Resource document. UC Berkeley, School of Information. Accessed 29 May 2021.
19. Micro Focus, Fortify Taxonomy: Software Security Errors. https://vulncat.fortify.com/en. Accessed 29 May 2021.
20. Basili, V., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., Seaman, C., Trendowicz, A., GQM+Strategies: A Comprehensive Methodology for Aligning Business Strategies with Software Measurement, 2014
21. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Experimentation in software engineering. *Springer Science & Business Media*. https://doi.org/10.1007/978-3-642-29044-2
22. Brooke, J. “SUS - A quick and dirty usability scale.”, in Usability evaluation in industry, CRC Press, 1996.
23. 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. https://doi.org/10.1007/s10339-009-0268-9
24. Baldassarre, M.T., Caivano, D., and Visaggio, G., “Software renewal projects estimation using dynamic calibration,” International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings., 2003, pp. 105-115, https://doi.org/10.1109/ICSM.2003.1235411.