Building GDPR-Compliant Web Applications with RuleKeeper

(extended abstract of the MSc dissertation)

Mafalda Baptista Ferreira
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisor: Professor Nuno Miguel Carvalho Santos

***Abstract*—Modern web applications provide many useful services to end-users which require them to blindly share their data. In 2018, the European Union issued the GDPR, a comprehensive legislation that defines a system of laws aimed at promoting the deployment of extensive security mechanisms for the protection of users' data and prevention of privacy breaches. Unfortunately, most modern systems tend to be optimized for performance, cost, and reliability, leaving security as a secondary goal. As a result, not only the web users remain prone to numerous risks, including the exposure of sensitive data, but the organizations themselves may incur high fees in the case of non-compliance with the GDPR. Considering these challenges, this thesis studies the implications that GDPR holds in web applications and clarifies the requirements organizations need to follow when managing their information systems. Particularly, this thesis presents RuleKeeper, a novel web application framework, tailored to provide data security and privacy protections according to GDPR-compliant policies.**

I. Introduction

The continuous technological growth and globalization have boosted the free flow of data and increased the scale of collection and sharing of personal data. Nowadays, people tend to share their data publicly and at an unprecedented scale to use many of the services provided by both private and public companies and authorities. Given that the protection of natural persons concerning the processing of personal data is a fundamental right [1][2], the need for establishing some data privacy and security standards has arisen. To protect citizens, the European Union has issued a regulation that lays a collection of rules and guidelines that aim to ensure the deployment of extensive security mechanisms for the protection of users' data and privacy – the *General Data Protection Regulation* (GDPR) [3]. As today many data processing and storage platforms are based on web applications, the existence of these mechanisms has an essential role in the data privacy and security regarding web applications because web users face numerous risks whenever they entrust their data to web applications. One of such risks consists in the exposure of sensitive data which is considered one of the most serious web application security risks since many applications and APIs do not properly protect sensitive data, such as financial, healthcare, and personally identifiable information. In fact, over the last few years, this has been the most common impactful attack on organizations [4]. Software exploits, weakly protected data, mishandling of server-side services, or questionable practices carried out by organizations are just a few examples of hazards that may lead to data breaches and loss of privacy. As a result, not only the web users remain prone to the aforementioned risks, but the organizations themselves may incur high fees in case of non-compliance with the GDPR.

However, building information systems as off now has been at odds with compliance regarding the GDPR. Frameworks have become an essential part of web development, providing not only scalability, integration, and robustness, but also being easy to integrate and time-saving. As these frameworks are not GDPR-compliant by default, application developers face numerous challenges in building their applications in adherence to the strict GDPR data protection policies. First, it is fundamental to understand what GDPR-compliance really means, addressing the challenges it raises and its ambiguities. Second, it must be stipulated which GDPR policies need to be enforced to ensure GDPR-compliance. Thirdly, the enforcement of GDPR policies must be assured without disrupting existing web development frameworks. The goal of our work is to tackle these problems and make the following central contributions.

The goal of our work is to build a web framework that enables web developers to write applications that can handle user data in compliance with GDPR policies while preserving the maintainability and performance delivered by the modern web frameworks. Such framework shall allow the specification of data protection policies in accordance with the GDPR and hard guarantee the enforcement of such policies throughout the entire application life cycle. It shall help to prevent privacy breaches, increase transparency, hand over control of data to its owner, and assist organizations in avoiding penalties due to noncompliance.

In this paper we first present a detailed study of the GDPR to determine its implications concerning personal data processing and the challenges it raises, namely ambiguities, indefinitions and omissions. We clarify the requirements organizations need to follow when managing their information systems. We notice that many of these requirements go beyond traditional access control or obligation policies, and report the need to specify the organization's privacy policy in a clear and concise way. Then, we propose RuleKeeper, a web development framework for 3-tier web applications which provides an API for the development of PDO-based

web applications and a system for enforcing compliance with GPSL policies. RuleKeeper enables web developers to write applications in compliance with GDPR policies while preserving the maintainability and performance delivered by modern web frameworks. We implemented a use case application to determine both the relevance, usability of our system, and the GDPR-compliance coverage, and also evaluated our system based on microbenchmarks. To promote its widespread adoption, RuleKeeper is based on some of the most popular tools used by web developers today, namely MongoDB, Express, React, and Node.js. We will make our system available.

II. Related Work

This section relates both GPSL and RuleKeeper's design to prior work, oriented to the GDPR compliance.

A. Existing Research on GDPR Compliance

Some recent studies aim to investigate the best practices and limitations in GDPR compliance and the extent to which the data protection policies prescribed by this regulation have been implemented in reality. The study by Mohan et al. [5] leverages on the importance of clear and concise privacy policies and proposes some recommendations for GDPR-compliant privacy policies supported by the principles laid out by the regulation. A recent study on Google's RTBF [6], highlights the challenges of implementing privacy regulations in practice, showing that, although users are exercising their rights, sometimes it is not a straightforward decision, for example, when the public interest outweighs the individual's right to privacy. The impact of the GDPR on storage systems [7][8] was also studied, illustrating the challenges of retrotting existing systems into compliance, the implications for system designers and the issues it introduces concerning performance. Schwarzkopf et al. [9] share a vision in how to achieve GDPR compliance in web service's backends, oriented towards data subject's rights and how to provide them control of their own data, resorting to cross-system abstractions and declarative specifications.

B. Security Policy Specification Languages

Much work has been done regarding security policy languages, XACML [10] is characterized for being a very expressive language concerning the access control domain, and PPL [11] and A-PPL [12] are more oriented towards privacy, extending XACML regarding privacy and accountability capabilities, respectively. This expressiveness has the cost of having a high specification complexity and verbosity, which makes them very user-unfriendly and decreases performance, unsuitable to being used by Data Protection Officers. Rego policies [13], are very simple, flexible, easy to use, and maintainable. Although it is not privacy oriented, its flexibility allows us to enhance it with the necessary extension points. It supports structured document models, such as JSON, a very valuable feature when building web applications.

C. Policy Enforcement Systems for Web Applications

Over the years, several security systems proposed mechanisms to allow the enforcement of end-to-end security policies for web applications, mostly employing information flow control and its variants or mandatory access control. Systems such as SIF [14], Fabric [15], or Hails [16] use IFC techniques for enforcing fine-grained security policies. However, these systems presented some major drawbacks. First, the security policies must be specified alongside the application code, which goes against our maintainability goal. Riverbed [17], also based on IFC but unlike previous systems, does not require developers to manually annotate code with labels or specify security policies alongside the application code. However, it only allows user-defined policies regarding their own personal data, meaning that a DPO is unable to specify the organization's security policy to be enforced by all users at the same time. As for mandatory access control systems, Qapla [18] provides fine-grained access control in database-backed application. Estrela [19] follows a different approach, leveraging some interesting techniques for enforcing contextual and granular policies concerning data protection regulations. However, both these systems provide poor policy management and maintainability, low usability by DPOs, and are also not compliant with the GDPR by default. OPA [20] provides a lightweight policy engine for a unified enforcement of Rego policies. However, it does not provide mechanisms to ensure proper compliance with all the GDPR requirements (such as data subject rights) and is not compliant with the GDPR by default. All these limitations have prompted us to develop RuleKeeper.

III. GDPR Implications on System Design

The General Data Protection Regulation (GDPR) [3] is the core of Europe's data privacy and security legislation. It came into force in 2018, and it sets a collection of rules regarding the protection of natural persons, respective to the personal data processing. As the infringement of the regulation can lead to high administrative fines, organizations are urged to seek compliance. Unfortunately, the integration of GDPR privacy requirements into information system brings several challenges, as some of the guidelines lie in grey areas and are up to interpretation. First, new actors are introduced to the system, such as the data subject, data controller, and data processor, urging the need to separate their responsibilities in the system. Second, GDPR introduces the concept of personal data. According to the regulation, personal data is any information relating to an identified or identifiable natural person all of the regulation applies to its processing. Since the data is stored in databases, without any standard or mandatory structure, information systems do not, by default, distinguish data type categories, such as personal data, sensitive data, health related data, or the others. With the conception of personal data, it is expected to associate the personal data to the corresponding data subject, that may or may not have consented to the processing of its data.

However, information systems process data, with no regard of who it belongs, and also query its databases without having the data subject into consideration. As so, there is no way to ascertain if the data subject that owns the data that is being processed consented to it or not. Lastly, information systems process and manage their data in accordance to their domain and goals, meaning that organizations cannot simply implement general purpose regulations, disregarding regulations which are domain-specific and defined by the organization. To mitigate these challenges for organizations, we lay down a set of principles that organizations must follow when managing their information systems. These principles are organized in accordance with their responsibility in the system: **C**X and **P**X concern the data controller and data processor responsibilities, respectively.

C01: Purpose Limitation
Art. 5 of the GDPR states that personal data should only be *≪collected for specific purposes and should not be processed in a manner that is incompatible with those purposes≫* It does not clarify if the purposes must be the same for all data subjects, or if it is possible to allow the processing of personal data for only one of many specific purposes.

C02: Data Minimization
Art. 5 of the GDPR states that personal data should be *≪adequate, relevant, and limited to what is necessary in relation to the purposes for which they are processed≫*. The data controller must specify which personal data item is required for each purpose.

C03: Lawfulness of Processing
Art. 5 and Art. 6 of the GDPR state that the data processing is lawful in when *≪the data subject has given its consent for one or more specific purposes, when it is necessary for the performance of a contract, or if it is relevant to the public interest≫*, amongst others, reflecting the lawfulness base. However, the performance of a contract (which is widely employed by organizations) is not associated with any specific purpose, making it seem like, when the data subject performs a contract, it is implicitly consenting to all purposes of data processing, which may not be the case and also limits the data subject's freedom.

C04: Transparency of Processing
Art. 5 states that personal data shall be *≪processed lawfully, fairly, and in a transparent manner in relation to the data subject≫*. Art. 12 tells that *≪the controller shall take appropriate measures to provide any information referred to in Arts. 13 and 14, and any communication under Arts. 15 to 22 and 34 in a concise, transparent, intelligible and easily accessible form, using clear and plain language≫*. It reflects the need of elaborating a clear, concise, and complete privacy policy that discloses the ways the organization gathers, uses, discloses, and manages personal data.

P01: Storage Limitation
Art. 5 states that personal data shall be *≪kept in a form which permits the identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed≫*. The data processor has to decide the point in time for marking data for deletion and erasing the expired data, which involves finding a sweet spot between a three-pronged trade-off: resource efficiency, performance, and full compliance. It also fails to state the conditions that make the data subjects prone to identification.

P02: Accuracy Preservation
Art. 5 states that personal data must be *≪accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that personal data that are inaccurate, having regard to the purposes for which they are processed, are erased or rectified without delay≫*. It does not define unequivocally which criteria should be adopted to validate the accuracy of personal data, which is highly dependent on the business case. It also omits the definition of specific deadline or criteria for the rectification/erasure of data if it is found to be inaccurate.

P03: Accountability
Art. 5 states that controller shall be *≪responsible for, and be able to demonstrate compliance with the regulation≫*. Art. 30 complements it by affirming that each controller/processor shall *≪maintain a record of processing activities under its responsibility≫*. Arts. 33 and 34 both state that when the personal data breach is *≪likely to result in a high risk to the rights and freedoms of natural persons, the controller shall communicate the personal data breach to the data subject and supervisory authority without undue delay≫*. This involves the deployment of an extensive and complex backend structure to support the monitoring of logs, and the GDPR does not describe the level of detail at which the monitoring activities must be carried out.

P04: Security of Processing
Art. 5 states that the personal data should be *≪processed in a manner that ensures its appropriate security, using appropriate technical or organizational measures≫*. Art. 32 complements it with measures that may be implemented to ensure the security of personal data. The data processor needs to deploy a cross-cutting security infrastructure that can mitigate the risks mentioned in both articles, which translates in a quite extensive wish-list and requires considerable investments from the organizations to keep the systems protected against constantly evolving security threats.

Data Subject Rights
Art. 12 states that the *≪controller shall facilitate the exercise of data subject rights under Arts. 15 to 22≫*. It is advised the controller to facilitate the exercise of the data subjects' rights, including the provision of mechanisms that the data subject can leverage to exercise its rights and, if applicable, obtain, free of charge, an appropriate response from the controller. Since the exercise of data subject rights depends on the organization's policy, organizations must specify in their policy how the system should implement such rights, stating which rights can be exercised automatically and which ones must go through the data controller for approval.

With this set of principles and challenges in mind, we

introduce RuleKeeper, a web application framework that allows the development of GDPR-compliant web applications.

IV. System Design

We present RuleKeeper, a web development framework that allows the development of GDPR-compliant web applications. In particular, it implements a new access control model that allows for the enforcement of declarative GDPR-aware policies.

A. Design Goals and Threat Model

The goal of RuleKeeper is to allow the specification of data protection policies in accordance with the GDPR and hard-guarantee the enforcement of such policies throughout the entire application lifecycle so as to help prevent privacy breaches, increase transparency, and hand over control of data to its owner. RuleKeeper is designed to be integrated into 3-tier information systems. In the design of our system, we consider the following main requirements: (i) **GDPR-policy compliance**: Personal data processing must be restricted according to a privacy policy that expresses constraints imposed by GDPR and the organization's DPO, (ii) **Maintainability**: The policy enforcement must be detached from the application code and allow for the separation of cross-cutting concerns and (iii) **Good end-to-end performance**: The incorporation of the necessary logic required for the validation and enforcement of policies should not significantly slow down the performance of applications.

We consider the existence of a dedicated person – the operator – which is responsible for the specification of the privacy policy to be enforced by a given web application. This policy is expected to reflect the specific terms of the GDPR regulation that apply to the concrete organization running the web application. We assume that the policy correctly expresses the data protection measures that are expected to be implemented. The web application developers write the code of their applications using the programming abstractions offered by our framework. This code is untrusted in the sense that the developer may accidentally introduce bugs and vulnerabilities in the application that can potentially lead to the violation of the privacy policy. As for the execution environment, we trust the correctness and integrity of the browser's runtime, the security of communication channels between client and server, and the correctness and integrity of the server-side OS, web server, and DBMS.

B. Architecture

RuleKeeper is a web development framework for 3-tier web applications which provides a system for enforcing compliance with GDPR data protection policies. Figure 1 illustrates a deployment of our system that supports the execution of a web application in the context of an organization. RuleKeeper system is composed of two key components: a *middleware* and a *manager service*. The middleware consists of a set of libraries linked to the web application, which export an API that allows the interaction between the web

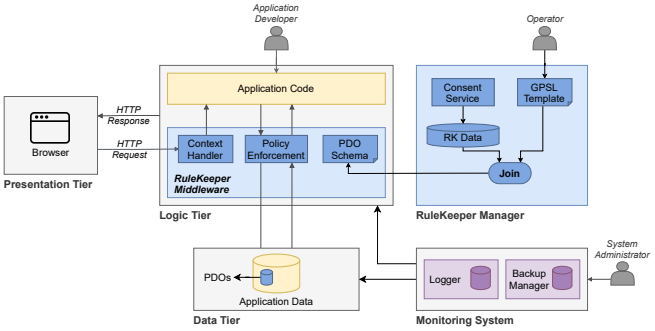

Figure 1: RuleKeeper architecture. Yellow boxes represent components specific to the application; blue boxes refer to RuleKeeper's components; and purple boxes pertain to external systems.

application and the GDPR policy enforcement system. The manager service runs in a centralized management server, and it is responsible for managing and coordinating the GDPR data protection policies, which it then shares with the middleware. The middleware uses these policies to enforce GDPR compliance as the web application collects and processes personal data.

There are two main actors that interact with RuleKeeper: the *application developer* and the *operator*. The application developer uses the middleware to develop and deploy GDPR-compliant web applications. Application development includes the specification of the data model, the implementation of the application operations, and the specification of an access control model for the principals that interact with the application. The operator is a person designated by the host organization which is responsible for managing the system and ensuring that the web application abides by the GDPR requirements. The operator receives input from the application developer concerning the web application's architecture (e.g., database model), and from the Data Protection Officer regarding the specific implementation of the GDPR for the organization.

RuleKeeper operates using Purposeful Data Objects (PDOs). PDO is a new abstraction that models the web application's state so as to satisfy GDPR restrictions. It includes data model properties, application code aspects, and the binding of such abstractions to the web application concerning the restrictions imposed by the GDPR, such as purposes and data minimization limitations. The system maintains a specification of this abstraction, the PDO Schema. This schema is built from two sources. The primary source is from a declarative policy specified by the operator in the GDPR Policy Specification Language (GPSL). GPSL is a simple domain specific language that we have developed to express GDPR policies for a given organization. The second source is some additional information that needs to be obtained at runtime, such as the data subjects' consent. The PDO abstraction allows the middleware to enforce the GDPR policies, without requiring the developer's direct

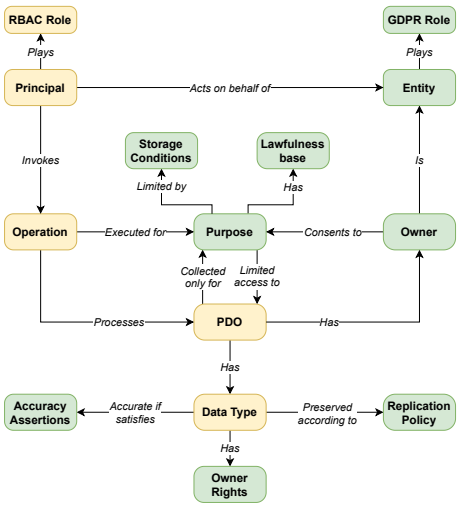

Figure 2: Purposeful Data Object model.

involvement.

C. Purposeful Data Objects

To accommodate such requirements when building information systems, we present a new abstraction, named Purposeful Data Objects (PDOs), which allows to model the web application's state, covering the required information for achieving GDPR policy-compliance, as represented in Figure 2. Some of the components presented in this model represent aspects that already exist in typical MVC web applications, and therefore are managed by the developer. These components are represented in the PDO model using yellow boxes. Green components represent the concepts introduced by the GDPR and are managed by the operator.

In an information system, there is a set of users that interact with the web application, which we name **principals**. Principals can have **roles** associated, representing their function and permissions to execute some operations in the system and their identification is done through their login. These principals can represent **entities** in the organization's domain, involved in the personal data processing. To each entity is attributed a specific **role** from the GDPR, describing their role in the personal data processing. Data subjects consent to one or more purposes regarding the processing of their data, and therefore are **owners** of their data. To ensure compliance with the GDPR regulations, information systems must facilitate the exercise of data subject rights. Organizations must specify in their policy how the system should implement the **owner rights**, which includes which data is accessible when fulfilling such right. PDOs are accessed and processed through system **operations**, which are executed to fulfill a specific purpose. As per the *Purpose Limitation* and *Data Minimization* principles, each PDO is collected for specific purposes and each purpose can only process a **limited** and relevant set of data. Then, each purpose must be associated with a **lawfulness base**, representing the valid lawful reason to process the personal data. Since principle *Storage Limitation* states that personal data should be erased when it is no longer necessary to the purposes for which the personal data was collected, purposes must also be associated with **storage conditions** that restrict the storing of the data collected for that purpose. Lastly, *Accuracy Preservation* mandates that this data must be kept accurate, which is assessed through **accuracy assertions**.

D. Application Development

The application developer is accounted for the implementation of the web application, which includes the specification of the data model, the implementation of the application operations and the specification of an access control model. This implementation is agnostic to GDPR abstractions. First, the web developer must specify how data of the application domain is organized in the database, by indicating the existing tables and its corresponding columns and data types. Following the specification of the application's data model, the developer must now implement the application operations' logic. Operations are the heart of the application as they are responsible for processing the application data and displaying those results to the end-user. Lastly, not all operations can be executed by all principles. The web application principals must authenticate themselves in the system, to express their role in the application, which represent their function and permissions to execute the application's operations. To support this model, RuleKeeper provides an authentication mechanism, enhanced with a session management api.

E. Policy Specification

Following the implementation of the web application, the operator is now responsible for writing a declarative policy that unifies the concepts of the PDO model, through a **GPSL template** provided as input to the RuleKeeper Manager. With this in mind, we present GPSL, a domain-specific policy specification language based on our PDO model. GPSL allows an operator to specify, in a non ambiguous way, the GDPR privacy requirements of the organization in such way that, when interpreted by the RuleKeeper system, the protection of personal data will be ensured to be in compliance with GDPR. We present its core language primitives and then elaborate on some future extensions to be developed in future work.

Abstract notation: GPSL can be used to specify the organization's privacy policy and the necessary mapping for providing RuleKeeper with application-dependent context.

$$\mathbb{A} : x_1 ... y_1, ..., x_n ... y_n$$

In this expression, $\mathbb{A}$ represents the PDO model property which GPSL describes and $x_i ... y_i$ represent its attributes. We proceed to specify the PDO model with this notation.

1) Core GPSL Primitives: We present the core GPSL primitives that are fully supported by the current version of RuleKeeper. These primitives are used as the basic language constructs for expressing GDPR-specific access control policies. Next, we enumerate each of these primitives.

Data Types: First we need to describe which data types d_i are classified as personal data. In a web application context, the data type identifier does not hold any value, so, each data type must be mapped to the corresponding table t_i and column c_i in the database.

$$\boxed{\mathbb{D} : d_1.(t_1.c_1), ..., d_n.(t_n.c_n)}$$

Purposes: To describe the purposes involved in personal data processing, we associate each purpose p_i with its lawfulness base b_i and the maximum data d_i they are allowed to process. If the lawfulness base is the explicit consent of a data subject or the execution of a contract, it is required the consent of the data subject.

$$\boxed{\mathbb{P} : p_1.b_1.d_1, ..., p_n.b_n.d_n}$$

Operations: PDOs are accessed and processed through system operations. Each operation o_i is executed with a purpose p_i and to which is only allowed the access to a subset of datatypes D_i. In a web application context, the operation identifier does not hold any value, so, each operation must be mapped to the corresponding url u_i in the web application.

$$\boxed{\mathbb{O} : o_1.p_1.D_1.u_1, ..., o_n.p_n.D_n.u_n}$$

Entities: The entities e_i involved in the personal data processing must have roles r_i associated, denoting their role in the GDPR: *data subject*, *data controller*, *data processor* or *third party*. Entities can either be declared statically in the policy, in the case of data controllers, processors and third parties, or can be mapped to a data table t_i column c_i, if they correspond to data subjects.

$$\boxed{\mathbb{E} : e_1.r_1, ..., e_n.r_n} \qquad \boxed{\mathbb{E}_{DS} : t_i.c_i}$$

Consent: If the entity represents a data subject o_i, it may consent to a set of purposes P_i regarding the processing of the DPOs it owns. The concept of data subject consent does not exist in the web application implemented by the developer. It is managed by RuleKeeper's **consent service**, and does not require mapping to the application.

$$\boxed{\mathbb{C} : o_1.P_1, ..., o_n.P_n}$$

Principals: Principals represent the users u_i that interact with the system, which can have roles r_i associated and can act on behalf of some entity e_i. Each role r_i is only allowed to execute a set of operations O_i in the system.

$$\boxed{\mathbb{U} : u_1.r_1.e_1, ..., u_n.r_n.e_n} \qquad \boxed{\mathbb{R} : r_1.O_1, ..., r_n.O_n}$$

Similar to the E_i entity policy, the association principal-entity can either be declared statically in the policy, in the case of data controllers, processors and third parties, or can be mapped in the database. In the last case, we define a policy that makes an association between the column p_i that identifies principal and the column e_i that identifies the entity, in a table t_i.

$$\boxed{\mathbb{U}_{DS} : t_i.e_i.p_i}$$

Data Ownership: Each PDO must be associated with its owner. As PDOs correspond to data stored in the database tables, we associate each table t_i that contains personal data with the column o_i of that table that identifies the owner of such data.

$$\boxed{\mathbb{N} : t_1.o_1, ..., t_n.o_n}$$

The normal activity of the system will be the invocation of operations that will interact with the data store. Each operation o will be allowed to be executed by the principal p if the following conditions are satisfied:

1) exists r in *roles*(p) such that $r \in$ *granted-acess*(o)
2) for all t in *typeset*(o), exists p_t in *purposes*(t), and exists p_o in *purposes*(o), such that $p_o \in p_t$
3) for all t in *typeset*(o), exists p_t in *purposes*(t), such that $t \in$ *maximum-data*(p_t)
4) exists p in *purposes*(o) and *requires-consent*(p) then, for all d in *dataset*(o), *granted-consent*(owner(d), d, p)

The condition (1) validates the access control. The condition (2) refers to purpose limitation and the condition (3) to data minimization requirements. Condition (4) refers to to lawfulness base requirements, where an operation that acts upon the PDOs for a given purpose and that purpose requires consent of the data subjects, it can only be executed if the PDO owners have granted its consent. The nomenclature used in these conditions is described as follows:

- *granted-access(o)*: roles authorized to perform operation o, specified by the principals' policy $\mathbb{U}$ and $\mathbb{R}$.
- *typeset(o)*: types of data processed by operation o, specified by the operations' policy $\mathbb{O}$.
- *purposes(o)*: purposes to which the operation o is executed, specified by the operations' policy $\mathbb{O}$.
- *maximum-data(p)*: the maximum data the purpose p is allowed to access, specified by the purposes' policy $\mathbb{P}$.
- *requires-consent(p)*: checks if the lawfulness base associated with the purpose p requires the data subject consent, specified by the purposes' policy $\mathbb{P}$.
- *dataset(o)*: PDOs processed by the operation o, specified by the operations' data type's policies $\mathbb{O}$ and $\mathbb{D}$.
- *granted-consent(o, d, p)*: checks if the owner o consented to the processing of its PDO d for purpose p, specified by the ownership policy $\mathbb{N}$, entities' policy $\mathbb{E}$ and consents' policy $\mathbb{C}$.
- *owner(d)*: data subject that owns the PDO d, specified by the ownership policy $\mathbb{N}$.

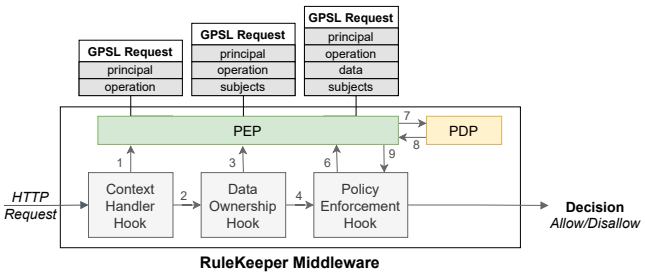

Figure 3: RuleKeeper's policy enforcement flow.

F. Policy Enforcement and Lifecycle

As presented in Figure 1, RuleKeeper Manager is responsible for storing and managing the GPSL template, which is managed by the organization's operator, and also for storing the RuleKeeper Data, which is the required data to evaluate the GPSL policies. The RuleKeeper Manager merges this information, generating a GPSL manifest, that is used by RuleKeeper's middleware to enforce such policies. RuleKeeper middleware is composed of three hooks that interact with RuleKeeper's PEP to convert the application request to a GPSL authorization request, evaluated by RuleKeeper's PDP, configured with the GPSL policies. The PEP is responsible for generating a GPSL authorization request and triggering the generation of an authorization decision in the PDP. Figure 3 describes the GPSL policy enforcement flow through arrows, representing the sequential interactions with both PEP and PDP.

Context Handler: The data protection policies' evaluation depends on the entity performing the operation and on the operation being executed, as different operations may have different purposes or different access levels. So, PEP generates a *GPSL Authorization Request* pre-filled with the context of the request: the principal executing the request and the operation being executed.

Data Ownership: To evaluate the GPSL policies, is required to know to which data subject the processed data belongs to. On top of that, requests to the web application may involve personal data belonging to several data subjects at the same time. PEP adds information to the GPSL Request regarding the data subjects involved in the request.

Policy Enforcement Hook: The policy enforcement hook receives the information generated by the previous hooks, and as it intercepts the database query, it turns this request into a complete GPSL authorization request by adding the requested data. It then uses the complete GPSL request to query the PDP, which returns the authorization decision generated by the GPSL data protection policies evaluation. In case of a positive authorization decision, the request is forwarded to the application controller, otherwise, the request is denied.

V. Implementation

We implemented a RuleKeeper prototype for MERN web applications. The RuleKeeper prototype comprises a middleware based on Node.js and Express, to be integrated in MERN web applications, and a management server, implemented as a MERN web server as well. We used Mongoose to help us model the MongoDB application data. These components communicate through web sockets, implemented with Socket.IO. We implemented RuleKeeper middleware as an external package, to incorporate it as seamlessly as possible, requiring minimal effort from the developer. RuleKeeper's policy enforcement is based on the interception of both the HTTP request and the subsequent database queries. To intercept the HTTP Request, we used the Express built-in application-level. Intercepting the database queries uses the same approach, but employing Mongoose middleware. We specified our GPSL policies using the Rego language and used Open Policy Agent as our policy engine. We integrated the OPA Policy Engine with the WebAssembly module to evaluate the policies.

VI. Case Study

Not only the regulation is extensive and considerably poor on specifics, but its implementation also depends on the organizations' domain. To evaluate the relevance and usability of RuleKeeper, we applied it to a concrete case study, allowing us to perform a real-world validation of our system. We collaborated with **LEB** - *Laboratórios Elisabete Barreto*, a clinical laboratory with the aim of developing a prototype intranet service for supporting its internal administrative processes.

A. GDPR support for health organizations

Whenever an organization seeks to be compliant with the regulation, the data controller must learn, interpret, and apply the regulation principles to their specific case. To simplify that process, some associations study and develop frameworks that determine the best approach for certain types of organizations. Thus, organizations can follow a certified framework, developed and optimized by a specialized association, without having to implement it from scratch, making the process less error-prone. APAC, a Portuguese association for clinical analysts who collaborates with national and international health institutions, developed a turnkey GDPR framework that highlights a collection of principles and establishes a set of technical and organizational measures for achieving GDPR-compliance, adopted by LEB.

B. Internal Process Analysis

To design and develop a web application that supports the internal administrative processes of the LEB organization, we performed a detailed and extensive analysis of the impact assessment reports regarding personal data processing for clinical analysis laboratories, developed by APAC, and the LEB's internal administrative processes, conducted by LEB. We selected the four LEB processes that process personal data and therefore need to be compliant with the regulation: (1) the pre-analytic process, (2) the analytic process, (3) the post-analytic process and (4) the human resources management process. The pre-analytic process is responsible

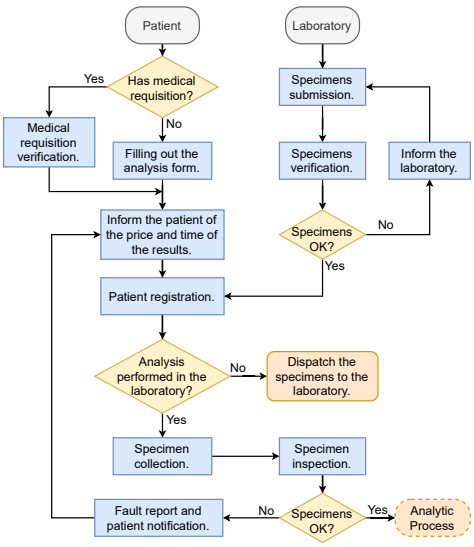

Figure 4: Flowchart of the LEB pre-analytic process. Decision blocks are represented in yellow, process blocks in blue and terminal blocks in orange.

for the patient registration and the specimen preparation and processing. This process is individually detailed in Figure 4. The analytic process is responsible for the specimen analysis and validation, the post-analytic process is responsible for preparing and emitting the analysis results and the human resources management process is responsible for hiring and training new employees. As all of these processes process personal data, APAC assigned a purpose to each one of them. For the clinical processes, the purpose is *Clinical Analysis*, and for the human resources management process, the purpose is *Human Resources*.

C. Prototype Implementation

Taking into consideration our analysis of the LEB internal administrative processes and their implications regarding personal data, we implemented a prototype intranet service, based on the MERN stack with an in-memory database. Our prototype supports three types of users: patients, receptionists, and system administrators. It simulates the actions between such users and the service, supporting the mentioned processes, specifically the following operations: (i) patient registration by receptionists, (ii) patient data handling by both receptionists and patients and (iii) user management by system administrators. We implemented eight controllers as described in Table II. To inform the data subjects on the processing made regarding their personal data, LEB composed a privacy policy and a document named *Information on the Processing of Personal Data*. Both documents disclose how LEB processes personal data and how it applies data protection principles, meeting **Arts. 12** (*Transparent Information*), **13**, and **14** (*Information to be provided*) of the **GDPR** [3]. To demonstrate that GPSL fully supports the policies detailed by LEB's privacy policy, we matched the LEB's privacy policy requirements with the GPSL policies, as depicted in Table I.

D. Portability Effort

The integration of RuleKeeper in our clinical analysis laboratory prototype consisted in integrating the RuleKeeper code in the LEB prototype web application and describing the organization's privacy policy in GPSL. To integrate RuleKeeper in the LEB web application, it was required to import and initialize RuleKeeper middleware. This involved adding 3 lines of code to the LEB web application. Then, RuleKeeper Manager requires a setup, involving setting up the RuleKeeper tables and mapping them to the data model. Lastly, we needed to integrate RuleKeeper management api calls to update the RuleKeeper Manager tables. This involved adding a total of 8 calls. Last and more challenging of all, we tried to describe LEB's privacy policy in the GPSL's Rego implementation. This task came out very complex and tricky since the LEB privacy policy was lacking important GPSL requirements.

VII. Evaluation

We demonstrate that RuleKeeper enforces GDPR-policy compliance, while providing good application maintainability features and delivering good performance results. We evaluate RuleKeeper's performance as part of the prototype implementation of our laboratory use case web application.

A. Methodology and Metrics

We evaluate RuleKeeper's performance evaluation aiming at measuring the performance overhead of RuleKeeper in terms of latency and throughput. To measure RuleKeeper's latency, we measured the total execution time of the system without RuleKeeper and the total execution time of the system with RuleKeeper. The throughput measurement was performed by saturating the system with and without RuleKeeper and calculating the maximum sustainable rate. To test requests that may result in different outcomes, we performed these tests using 3 different controllers. For that we describe 3 request types, characterized by parameters that may influence RuleKeeper's performance: the role of the entity performing the operation and the number of data subjects involved. We then associate each one of them with a controller implemented in the use case application, as described in Table III.

Our testbed is composed of five 64-bit Ubuntu 18.04.5 LTS virtual machines (VMs) provisioned with 20GB of RAM and eight virtual Intel Xeon E5506 2.13GHz CPUs, located in different physical machines but connected over a local network. VM1 runs the use case application, extended or not with RuleKeeper, depending on the test, and VM4 runs an in-memory MongoDB database with the use case application data. VM2 executes an instance of the RuleKeeper Manager prototype and VM3 runs an in-memory MongoDB database with the RuleKeeper Manager data. Finally, VM5 is used to run the client-side experiments with up to 50 client instances, running simultaneously to saturate RuleKeeper.

LEB Privacy Requirement	GPSL Policies
'Personal data will be exclusively processed for the clinical analysis activity exercise, by Elisabeth Barreto Laboratories"	Entities $\mathbb{E}$ policy: 'Laboratory LEB'.'data controller'
'The purposes of the data collected are: Management of information regarding clinical analysis services, Human resource Management, Marketing (if applicable) and Video surveillance, (if applicable)", '*The legal basis for the processing of your personal data is the contract established at the time you ask us to perform clinical analysis.*"	Purposes $\mathbb{P}$ policy: 'clinical analysis'.'execution of a contract', 'human resources management', 'marketing', 'video surveillance'
On the first visit to the laboratory, the patient is asked for the citizen card to open the identification form and the following data can be collected: full name, date of birth, sex, beneficiary number, TIN, photo, address, mobile phone/telephone, email and relevant clinical observations.	Data Type $\mathbb{D}$ policy : (full name, date of birth, sex, beneficiary number, TIN, photo, address, mobile phone, clinical information)'clinical analysis'.'patient A' Consent $\mathbb{C}$ policy : 'patient A'.'clinical analysis'
The collected data is processed and stored computerized (...) during the minimum period necessary for use according to the purpose for which they were collected	This data is insufficient to specify a storage limitation $\mathbb{S}$ policy.
The data collected and held by LEB — Elisabeth Barreto Laboratories may be transmitted (...) to the following entities: Health Insurance and Subsystems; Health professionals; Subcontractors who will process the data on behalf of LEB — Elisabeth Barreto Laboratories and according to the purposes determined by it.	Entities $\mathbb{E}$ policy: 'health insurance A'.'third party', 'subcontractor A'.'data processor'.
To request the exercise of any data subject right, must be sent an e-mail to *dpo@lebanalises.com*	Owner Rights $\mathbb{W}$ policy: 'right to access'.'indirect access mode'

Table I: LEB privacy requirements matched with GPSL policies.

Controller	Description
Register Patient	The receptionist can register new patients in the system.
Get Patient Information	The receptionist can fetch a patient's data. A patient can fetch its own data.
Get Information from all Patients	The receptionist can fetch data from all patients in order to query some parameters.
Update Patient Information	The receptionist can update a patient's data. A patient can update some aspects of its own data.
Register User	A system administrator can register a new user in the system.
Delete User	A system administrator can delete a user in the system.
Update User	A system administrator can update a user in the system.

Table II: Controllers implemented by the LEB prototype intranet service.

	Controller	Entity	Data Subjects
1	Access own patient data *GET /patients/:patientId*	Data Subject	1
2	Access patient data *GET /patients/:patientId*	Controller	1
3	Access 100 patient data *GET /patients/all*	Controller	100

Table III: Controllers used for performance measurements.

B. Execution Time Overheads

We simulated traffic by generating 10.000 sample requests sequentially for each one of the 3 controllers and measuring the time required to return each response. For each controller, we measured the total execution time of the controller without importing RuleKeeper and the total execution time of the controller with RuleKeeper, reporting the arithmetic mean and 99th percentile. The execution time does not include web page loading and rendering.

Table IV presents the execution time overheads and Figure 5 show our results, where the yellow bars represent the controller execution time without RuleKeeper and blue bars represent the controller execution time with RuleKeeper.

	Controller Time (ms)		RuleKeeper Time (ms)		Overhead
	mean	99^{th}	mean	99^{th}	
#1	9.16	16.31	10.16	16.73	1.00 ms (9.84%)
#2	9.37	19.60	10.68	30.82	1.31 ms (12.27%)
#3	62.73	145.14	69.47	159.82	6.47ms (9.70%)

Table IV: RuleKeeper overheads, sampled from the generated traffic.

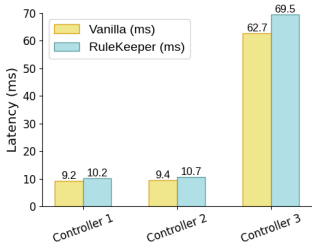

Figure 5: RuleKeeper overhead for 3 controllers in the use case application. Labels give the execution times.

On average, RuleKeeper overheads are low and likely unnoticeable to web application users: 8-12%. As expected, these are small client-perceived overheads. The overhead does not change with the type of database operation (read/update) since both types of requests pass through the same hooks and the logic contained in the Database Update hook produces a minimal overhead.

C. Operation Scalability

To evaluate RuleKeeper throughput under high load, we used wrk2 to generate sample requests of the 3 controllers and measured the maximum sustainable rate, i.e the maximum number of requests our server could respond to, before it could no longer answer incoming requests within a reasonable amount of time. We report the average operations/second, as observed in Figure 6, where listed areas represent the web application using RuleKeeper. In average, RuleKeeper responds to 20-30% fewer requests, regardless of the number of data subjects involved in the operation, but with worse throughput when the entity performing the action is not the data subject. Regarding the context of RuleKeeper applications, the provided throughput is still very acceptable to web application requests.

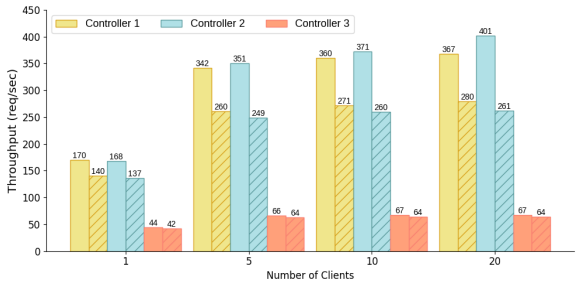

Figure 6: Throughput measured using the pre-defined LEB use case controllers.

D. Programming Effort

RuleKeeper prototype implementation is composed by the *RuleKeeper Middleware* to be imported by the 3-tier web applications, and the *RuleKeeper Manager* server. First, to use the RuleKeeper middleware in a Node/Express 3-tier application, the developer only needs to add 3 lines of code: importing the RuleKeeper libraries and integrating both the mongoose and express middlewares. It is also mandatory to setup and boot the RuleKeeper Manager and the RuleKeeper Manager database. So, the developer needs to create the MongoDB database with an empty collection - *consent*, import the code and define the mongodb database url. Finally, the data model specified by the developer must have, for each table that contains personal data, a column that identifies the data owner to support the data ownership mechanism.

VIII. Conclusions

In this thesis, we provided a detailed GDPR analysis to fully understand the challenges of GDPR compliance in information systems and presented a set of principles that organizations must follow when managing their information systems. Considering this set of principles and challenges, we designed and implemented RuleKeeper, a novel web application framework tailored to provide data security and privacy protections according to GDPR-compliant policies. RuleKeeper includes a declarative policy specification language, GPSL, that allows to specify, in a non-ambiguous way, the organization's privacy policies, based on a novel abstraction named *Purposeful Data Objects*. The PDO abstraction allows the application state modeling, covering the required information to achieve GDPR-compliance and supporting our GDPR analysis results. We collaborated with **LEB** - *Laboratórios Elisabete Barreto* to perform a real-world validation of our system, including the expressiveness of GPSL policies. The experimental evaluation conducted over RuleKeeper shows that the integration of RuleKeeper in existing web applications only adds small client-perceived overheads, while providing good application maintainability.

References

[1] The Member States, "Charter of Fundamental Rights of the European Union," *Official Journal of the European Union*, vol. C 326, October 2012.

[2] ——, "Consolidated version of the Treaty on the Functioning of the European Union," *Official Journal of the European Union*, vol. C 326, October 2012.

[3] The European Parliament and the Council of the European Union, "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 (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L 119, May 2015.

[4] OWASP, "Top 10 - 2017: A3 sensitive data exposure," https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure Accessed: 2020-12-10.

[5] J. Mohan, M. Wasserman, and V. Chidambaram, "Analyzing GDPR compliance through the lens of privacy policy," *CoRR*, vol. abs/1906.12038, 2019.

[6] T. Bertram, E. Bursztein, S. Caro, H. Chao, R. C. Feman, P. Fleischer, A. Gustafsson, J. Hemerly, C. Hibbert, L. Invernizzi, L. K. Donnelly, J. Ketover, J. Laefer, P. Nicholas, Y. Niu, H. Obhi, D. Price, A. Strait, K. Thomas, and A. Verney, "Five years of the right to be forgotten," in *Proceedings of the Conference on Computer and Communications Security*, 2019.

[7] A. Shah, V. Banakar, S. Shastri, M. Wasserman, and V. Chidambaram, "Analyzing the impact of gdpr on storage systems," in *Proceedings of the 11th USENIX Conference on Hot Topics in Storage and File Systems*, ser. HotStorage'19. USENIX Association, 2019.

[8] S. Shastri, M. Wasserman, and V. Chidambaram, "The seven sins of personal-data processing systems under gdpr," in *Proceedings of the 11th USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'19. USENIX Association, 2019.

[9] M. Schwarzkopf, E. Kohler, M. F. Kaashoek, and R. Morris, "Position: Gdpr compliance by construction," in *Poly/DMAH@VLDB*, 2019.

[10] OASIS, "eXtensible Access Control Markup Language (XACML) version 3.0," 2013, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf Accessed: 2020-12-10.

[11] C. Ardagna, L. Bussard, S. De, C. Vimercati, G. Neven, S. Paraboschi, E. Pedrini, F.-S. Preiss, D. Raggett, P. Samarati, S. Trabelsi, and M. Verdicchio, "Primelife policy language," January 2009.

[12] M. Azraoui, K. Elkhiyaoui, M. Önen, K. Bernsmed, A. S. De Oliveira, and J. Sendor, "A-PPL: an accountability policy language," in *Data privacy management, autonomous spontaneous security, and security assurance*, 2014, pp. 319–326.

[13] C. N. C. Foundation, "Rego Policy Language," 2019, https://www.openpolicyagent.org/docs/latest/policy-language Accessed: 2020-12-07.

[14] S. Chong, K. Vikram, and A. C. Myers, "Sif: Enforcing confidentiality and integrity in web applications," in *Proceedings of USENIX Security Symposium on USENIX Security Symposium*, 2007, pp. 1–16.

[15] J. Liu, M. D. George, K. Vikram, X. Qi, L. Waye, and A. C. Myers, "Fabric: A platform for secure distributed computation and storage," in *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles*, 2009, pp. 321–334.

[16] D. B. Giffin, A. Levy, D. Stefan, D. Terei, D. Mazières, J. C. Mitchell, and A. Russo, "Hails: Protecting data privacy in untrusted web applications," in *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*, 2012, pp. 47–60.

[17] F. Wang, R. Ko, and J. Mickens, "Riverbed: Enforcing user-defined privacy constraints in distributed web services," in *Proceedings of the USENIX Conference on Networked Systems Design and Implementation*, 2019, pp. 615–629.

[18] A. Mehta, E. Elnikety, K. Harvey, D. Garg, and P. Druschel, "Qapla: Policy compliance for database-backed systems," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1463–1479.

[19] A. Bichhawat, M. Fredrikson, J. Yang, and A. Trehan, "Contextual and granular policy enforcement in database-backed applications," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, p. 432–444.

[20] C. N. C. Foundation, "Open Policy Agent (OPA)," 2019, https://www.openpolicyagent.org/ Accessed: 2020-12-07.