Poster: A Systems Approach to GDPR Compliance-by-Design in Web Development Stacks

Mafalda Ferreira
INESC-ID & Instituto Superior Técnico,
Universidade de Lisboa
mafalda.baptista@tecnico.ulisboa.pt

Tiago Brito
INESC-ID & Instituto Superior Técnico,
Universidade de Lisboa
tiago.de.oliveira.brito@tecnico.ulisboa.pt

José Fragoso Santos
INESC-ID & Instituto Superior Técnico,
Universidade de Lisboa
jose.fragoso@tecnico.ulisboa.pt

Nuno Santos
INESC-ID & Instituto Superior Técnico,
Universidade de Lisboa
nuno.m.santos@tecnico.ulisboa.pt

ABSTRACT

Pressured by existing regulations such as the EU GDPR, online services must advertise a personal data protection policy declaring the types and purposes of collected personal data, which must then be strictly enforced as per the consent decisions made by the users. However, due to the lack of system-level support, obtaining strong guarantees of policy enforcement is hard, leaving the door open for software bugs and vulnerabilities to cause GDPR-compliance violations. We present ongoing work on building a GDPR-aware personal data policy compliance system for web development frameworks. Currently prototyped for the MERN framework, our system allows web developers to specify a GDPR manifest from which the data protection policy of the web application is automatically generated and is transparently enforced through static code analysis and runtime access control mechanisms. GDPR compliance is checked in a cross-cutting manner requiring few changes to the application code. We evaluate our prototype with four real-world applications. Our system can model realistic GDPR data protection requirements, adds modest performance overheads to the web application, and can detect GDPR violation bugs.

CCS CONCEPTS

• **Security and privacy** → **Web application security**; **Software security engineering**.

KEYWORDS

GDPR compliance, privacy policies, web security

ACM Reference Format:
Mafalda Ferreira, Tiago Brito, José Fragoso Santos, and Nuno Santos. 2022. Poster: A Systems Approach to GDPR Compliance-by-Design in Web Development Stacks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3548606.3563521

1 INTRODUCTION

Protecting personal data has become a crucial concern for most organizations with an online presence. In particular, the EU General Data Protection Regulation (GDPR) imposes strict access control requirements on data controllers when managing the personal data of their customers. To this end, data controllers must publish a human-readable policy declaring the personal data to be collected and the purposes for which it will be processed. Once explicit consent is granted by the data subjects, data controllers must strictly comply with the agreed policy, otherwise, they may incur the payment of heavy fines.

However, it is not easy to enforce personal data protection policies in a full-blown web application. Web developers often write their applications using full-stack development frameworks that are currently agnostic to the GDPR, such as MERN[1]. However, MERN and other popular web frameworks provide no native support to help web developers specify and enforce personal data protection policies, opening the door to GDPR compliance violations.

To help web developers fulfill the GDPR requirements for personal data protection, various system-level mechanisms have been recently proposed [3, 5, 7–10, 12–14]. On the one hand, consent management platforms [9] focus on the application frontend, enabling cookie banners to be transparently managed in a GDPR-compliant manner, but lack the mechanisms to enforce data protection policies at the application backend. In contrast, other systems [10] act at the storage layer alone, rendering them unable to detect data access violations that depend on the application context, e.g., the notion of purpose or user authentication state. Riverbed [12] and PrivGuard [13] adopt a truly holistic approach, where they keep track of personal data ownership and user consent preferences throughout the entire data workflow. However, these systems focus on specialized usage scenarios, relying on mechanisms (e.g., trusted hardware) which are difficult or unsuitable to apply in typical 3-tier web applications built with full-stack frameworks, which are the focus of our work.

2 GOALS

Our goal is to design a GDPR-aware policy enforcement system for web frameworks that can help web developers mitigate GDPR violations caused by application bugs and security vulnerabilities

[1]**MERN** stands for **M**ongoDB, **E**xpress.js, **R**eact.js and **N**ode.js.

that result in data breaches, such as SQL injections. Given the extensiveness of the GDPR, we concentrate specifically on ensuring compliance with the three GDPR guidelines: *purpose limitation*, *data minimization* and *lawfulness of processing*. We believe our solution can be extended to cover other GDPR requirements, which we leave for future work. Concerning the security vulnerabilities, our focus is to reduce the attack surface of vulnerable code, restricting its privileges to protection domains based on data access purposes. We look for a solution that: i) allows the specification of *expressive data protection policies*, being able to accommodate the needs of various organizations and application scenarios, ii) is *transparent*, ensuring that policies are shown to the users in a clear, complete, and accurate way, reflecting how the application actually handles the data, and iii) is *easy to maintain*, detaching policy enforcement from application code.

3 PROPOSED SOLUTION

We propose a GDPR-aware personal data compliance system for full-stack web frameworks. Using MERN for demonstration, it incorporates cross-cutting GDPR compliance features whilst requiring minimal changes to the application code. With our system, the data protection policy of the website is automatically generated based on a machine-readable GDPR *manifest* written by the web developer. This manifest specifies the personal data types and purposes for which the web application can process the data. The system will then strongly enforce the policy consent decision of website users, preventing business-level operations from manipulating personal data for purposes that data subjects (i.e., the users) have not agreed to. Put simply, by approving the data protection policy of a website powered by our solution, "what you agree to is what you get".

Our system design includes three main novelties that solve non-trivial challenges to automatically detect and prevent GDPR compliance violations. First, to overcome a semantic gap between abstract GDPR concepts such as "personal data" or "purpose" and the application's JavaScript code and MongoDB queries, it introduces a *domain-specific language* (DSL) for specifying the GDPR manifest. Our DSL features a small set of intuitive language constructs that allow developers to easily bridge this semantic gap without overwhelming them with unnecessary complexity or legal terminology.

Second, to prevent GDPR violations, the JavaScript code implementing the business logic and the database queries must not be allowed to process personal data unless this data is strictly used for the purposes indicated in the GDPR manifest. To perform these checks as the application codebase and data protection policy itself evolve, our system includes a *static analysis tool* that automatically checks for the presence of GDPR compliance bugs. This tool generates a graph-based model of the JavaScript code and uses it along with the GDPR manifest to look for violations of GDPR's purpose limitation and data minimization guidelines.

Third, the system needs to efficiently keep track of consent preferences for every user visiting the website and block any business-level operations that may attempt to access personal data against their will. Given that these access control decisions must be made dynamically, it includes dedicated middleware which implements the novel idea of *sticky banners*, i.e., an evolution of cookie banners

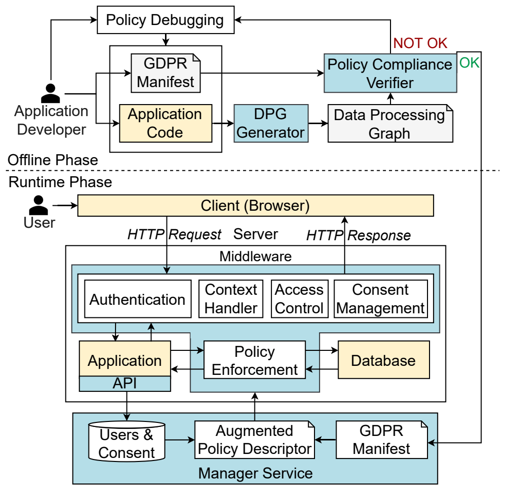

Figure 1: System architecture.

where users' consent is recorded and their respective policy decisions are dynamically enforced at runtime. We show that dynamic policy enforcement adds another defensive barrier against external attempts to exfiltrate personal data through the exploitation of vulnerabilities in the application code.

Figure 1 illustrates a deployment of our system in a 3-tier web application. The yellow boxes represent the application-specific components that are present in a typical behavior of a 3-tier architecture system. The blue boxes represent the specific software components of our system. It operates in two phases: the *offline* phase and the *runtime* phase.

The offline phase takes place at development time before deploying the web application. In this phase, the web developer specifies a GDPR *manifest* from which the data protection policy will be generated. Our system implements a static code analysis pipeline that allows the web developer to verify if the policy reflects the way that the application behaves. First, a code analysis tool generates a model of the application code which we designate Data Processing Graph (DPG). This model is then fed to a compliance verification tool that looks for inconsistencies between the DPG and the GDPR manifest. If so, the developer needs to debug either the application code or the GDPR manifest to ensure that there is a match between them. When the validation step passes, the GDPR manifest is loaded to the runtime components.

In the runtime phase, our system implements dynamic policy enforcement and consent management functions. Two components take charge of this phase: *middleware* and *manager service*. The middleware consists of application-linked libraries and plays two roles: i) keeps the manager server updated with user-related information, and ii) enforces dynamic access control decisions and consent management validations. The manager service runs in a centralized server and coordinates the middleware. Importantly, it generates a data structure named Augmented Policy Descriptor (APD) which contains the information required by the middleware to dynamically enforce the policy, such as the GDPR manifest and user consent preferences. The middleware leverages this information on two main occasions. In one case, it intercepts the HTTP

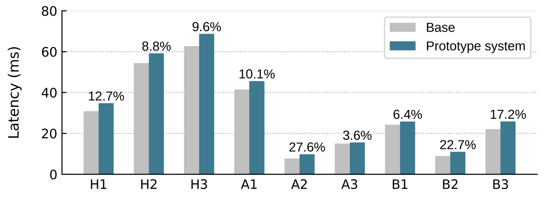

Figure 2: Average client-perceived latency in legacy application tasks. Labels show the relative overhead, in percentage.

requests to verify the user's credentials and consent preferences, and enforce access control accordingly. The middleware also intercepts the database queries of the application to validate if they satisfy the GDPR policy and block them otherwise.

To specify the GDPR manifest in an expressive and unambiguous way, a central challenge is to bridge the semantic gap between application and GDPR domains. Our approach is to create a simple DSL where the developer first specifies abstractions for two independent conceptual planes (application and GDPR) and then establishes mappings between them. On the application plane, the developer specifies the GDPR-agnostic attributes of the web application, such as the data types of the database schema and REST API operations. The GDPR plane describes attributes related to the GDPR requirements, including the personal data, the purposes, and which data is allowed to be collected for a given purpose. Although there are related DSLs for modeling and verifying systems' security properties [11], the provided abstractions do not easily map to our problem domain. This is a limitation that we overcome in our work.

4 PRELIMINARY RESULTS

We implemented an open-source prototype of our system for MERN. To assess the expressiveness power of its policy language and its fitness for real-world scenarios, we conducted four case studies. The first one was based on a real clinical analysis scenario, building the entire application from scratch, where we learned that it can be used to fully express data protection requirements in the health domain. We collaborated with the clinical laboratory *LEB - Laboratórios Elisabete Barreto* [1] to develop a prototype intranet service for supporting its internal business processes. The others are based on large preexisting and popular applications, and allowed us to learn that our system can detect compliance bugs in complex and evolving applications. We selected three open-source web applications: Habitica [6], a task management application with over 1M downloads in Google Play and 9k GitHub stars, Amazona [2], an e-commerce application, amazon style, with 1.3k GitHub stars, and Blog [4], a blog application with 3.3k GitHub stars.

We experimentally evaluated the performance of our prototype, and observed a 13.2% increase in average latency in legacy applications, from the perspective of Web clients. Figure 2 presents the average client-perceived latency of the legacy applications with (blue bars) and without our system (grey bars). In general, these overheads are dominated by the execution time of the policy enforcement module, which takes charge when intercepting the database queries. We consider the overheads incurred by the benchmark applications are acceptable given the added security and policy compliance benefits of our system.

5 CONCLUSIONS AND FUTURE WORK

This work presents a GDPR-aware policy compliance system designed for full-stack web development frameworks. It includes a policy specification language for specifying the GDPR requirements of the application and employs static and dynamic analysis to enforce policy compliance. We prototyped our system using MongoDB, Express.js, and Node.js and evaluated it with four case studies. Our system is able to prevent various GDPR compliance violations while registering acceptable performance overheads. As for future work, we intend to improve on some existing limitations, namely: i) extend the verified GDPR guidelines, ii) improve the accuracy of static analysis and characterize its soundness and completeness guarantees, iii) optimize the performance of our runtime enforcement engine, e.g., through query rewriting, iv) perform a usability study of our system, and v) strengthen our threat model by developing new defenses against explicit attacks to the system itself. For instance, an attacker may try to subvert the system's policy enforcement mechanisms by taking advantage of a software bug in the middleware. To mitigate these attacks, we propose to study the use of software verification techniques to preemptively detect bugs in the middleware that can lead to critical security breaches.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments and insightful feedback. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) via the 2021.06134.BD grant, and the UIDB/50021/2020, PTDC/CCI-COM/32378/2017 (INFOCOS), and CMU/TIC/0053/2021 (DIVINA) projects.

REFERENCES

[1] Análises Clínicas LEB - Laboratórios Elisabeth Barreto. 2022. Retrieved January 14, 2022 from https://www.leb-analises.com/.
[2] Basir. 2020. Amazona - Build ECommerce Website Like Amazon. Retrieved April 13, 2022 from https://github.com/basir/node-react-ecommerce.
[3] Abhishek Bichhawat, Matt Fredrikson, Jean Yang, and Akash Trehan. 2020. Contextual and Granular Policy Enforcement in Database-Backed Applications. In *AsiaCCS'20*.
[4] gothinkster. 2018. Blog - RealWorld Example App. Retrieved April 13, 2022 from https://github.com/gothinkster/node-express-realworld-example-app.
[5] Marco Guarnieri, Musard Balliu, Daniel Schoepe, David Basin, and Andrei Sabelfeld. 2019. Information-Flow Control for Database-Backed Applications. In *EuroS&P'19*.
[6] HabitRPG. 2021. habitica - Release v4.189.0. Retrieved January 14, 2022 from https://github.com/HabitRPG/habitica/releases/tag/v4.189.0.
[7] Rishabh Khandelwal, Thomas Linden, Hamza Harkous, and Kassem Fawaz. 2021. PriSEC: A Privacy Settings Enforcement Controller. In *USENIX Security'21*.
[8] Nico Lehmann, Rose Kunkel, Jordan Brown, Jean Yang, Niki Vazou, Nadia Polikarpova, Deian Stefan, and Ranjit Jhala. 2021. STORM: Refinement Types for Secure Web Applications. In *OSDI'21*.
[9] Célestin Matte, Nataliia Bielova, and Cristiana Santos. 2020. Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework. In *S&P'20*.
[10] Aastha Mehta, Eslam Elnikety, Katura Harvey, Deepak Garg, and Peter Druschel. 2017. Qapla: Policy compliance for database-backed systems. In *USENIX Security'17*.
[11] Tamjid Al Rahat, Yu Feng, and Yuan Tian. 2019. OAUTHLINT: An Empirical Study on OAuth Bugs in Android Applications. In *ASE'19*.
[12] Frank Wang, Ronny Ko, and James Mickens. 2019. Riverbed: Enforcing User-defined Privacy Constraints in Distributed Web Services. In *NSDI'19*.
[13] Lun Wang, Usmann Khan, Joseph Near, Qi Pang, Jithendaraa Subramanian, Neel Somani, Peng Gao, Andrew Low, and Dawn Song. 2022. PrivGuard: Privacy Regulation Compliance Made Easier. In *USENIX Security'22*.
[14] Sebastian Zimmeck, Rafael Goldstein, and David Baraka. 2021. PrivacyFlash Pro: automating privacy policy generation for mobile apps. In *NDSS'21*.