Impact of GDPR on Service Meshes

Ghulam Murtaza
ghulam_murtaza@brown.edu

Amir R. Ilkhechi
ilkhechi@brown.edu

Saim Salman
saim@brown.edu

Abstract

Through a swift migration that took place in early 2000s, many companies moved their data and services to clouds. This rapid adoption allowed little opportunity for the researchers to study the downsides of such massive shift in terms of privacy. Therefore, soon after the change, companies started to suffer detrimental data breaches. As a response, the European Union passed some regulations (e.g., GDPR) to mandate privacy requirements on companies handling data related to EU citizens. As a result, the companies started to abide with the rules by changing their widely deployed systems. Such a change is still on progress and requires massive engineering efforts because of the design issues inherent within the old systems. Another shift that we are witnessing is a widespread adoption of Service Meshes. In this project, we argue that using the standard tools that a service mesh manager such as Istio provides, it is possible to satisfy at least some of the most important rules mandated by GDPR.

1 Introduction

The introduction of cloud computing led to a tremendous shift in data storage, processing, and manipulation. Compared to the traditional systems, this novel paradigm not only offered a superior level of optimization, utilization, flexibility but also it was more scalable which attracted a large number of IT companies.

Soon after the commercial adoption by companies such as Amazon in early 2000s, a booming "cloud-based" trend started to be formed. The shift was so swift that the potential disadvantages of cloud systems were completely overlooked.

Among the most concerning threats faced by cloud-based companies is the breach of their (and therefore their users') private data. Moreover, among the most important examples of data privacy disasters, we can find are: 2013-14 Yahoo (3 billion user accounts) [10], 2014-18 Marriott International (sensitive data of 500 million customers) [9], and 2014 eBay (sensitive data of 145 million users) incidents [2].

As a response, the European Union passed Data Protection Directive (DPD), which went into effect in 1995. More recently, General Data Protection Regulation (GDPR) replaced DPD with a promise of a better level of privacy protection for European citizens [5]. To be more specific, GDPR is a regulation that mandates businesses to protect the personal data and privacy of EU citizens. Companies failing to comply with GDPR, could face a hefty fine.

The introduction of GDPR has brought about changes in the practices in cloud management both within and outside of EU. Indeed, even companies located in non-EU territories process EU citizens' data and they should all comply just like companies located in EU. Therefore, what we are particularly interested in is the impacts of GDPR on applications.

The introduction of the GDPR has led to companies tailoring their production level to the needs of the GDPR. For example, Facebook is going through such a process which is led by a Dublin-based data protection team. According to Facebook, it is the largest cross-functional team in Facebook's history [4]. Amazon is another example that promises GDPR compliance through mechanisms such as encryption, monitoring and logging, and access control among many other [**?**].

As companies hasten to modify their systems to make them GDPR compliant to not have to pay hefty fines (companies can be made to pay 4% of their annual revenue [5]), this is costing them huge amount of time and effort.

We noted that as companies are going through changes due to the GDPR, another cause for change has been due to micro-services and service meshes. Recently, companies have been converting their monolithic applications into micro-services. Micro-services are a different development method for applications and it provides benefits over monolithic applications largely in maintainability, test-ability.

Although, micro-services have their own benefits, its not a silver bullet and they come with their own drawbacks. To alleviate some of the drawback, service meshes have been invented. Service Meshes provide multiple features to micro-services: fine-grained traffic engineering, state-of-the-art security (mTLS) and logging + authentication systems [6].

We propose that service meshes can be easily utilized to trivially satisfy at least some GDPR requirements using only the tools that they provide for us without a need to implement complex modules on top of the existing systems. In summary, we make the following contributions:

1. We identify the service mesh features that would allow us to cater to the list of GDPR articles.
2. We make a application GDPR compliant and analyze the performance degradation caused by the various features.
3. We suggest new features that would reduce the overhead to make service mesh based applications GDPR compliant.

2 Background

2.1 Service Meshes

The development community shifting away from monolithic application design to a microservice-based architecture has given the need for service meshes. Service meshes help developers by hiding the complexity and scale of deployment and management of microservice-based architectures.

Service Meshes, considered by many as an orchestration layer, is architectured similar to SDNs i.e. it has a data plane and control plane. In Figure-1, one can see that the data plane consists of a series of L7 proxies where the proxies can be configured by the control plane. This allows developers unparalleled functionality in securing, monitoring and configuring their applications. The most common services meshes: Istio [6], Linkerd [8], Kuma [7] provide a basic set of functionality to developers: (i) mTLS, (ii) service discovery, (iii) traffic engineering functionality and (iv) monitoring + distributed tracing service.

2.1.1 Istio Architecture

In this paper, we've focused specifically on the Istio service mesh, shown in Figure-1. The Istio architecture consists of multiple components (divided into the control and data plane):

1. Control Plane
   (a) **Mixer** has a two-fold purpose: (i) it implements various types of policies (mainly security policies) and (ii) it gets telemetry data from the data plane proxies
   (b) **Pilot** is mainly used to configure the proxies to facilitate traffic engineering capabilities.
   (c) **Citadel** is mainly used for key and certificate management.
   (d) **Galley** is used to insulating the rest of the Istio components from the details of obtaining user configuration from the underlying platform.

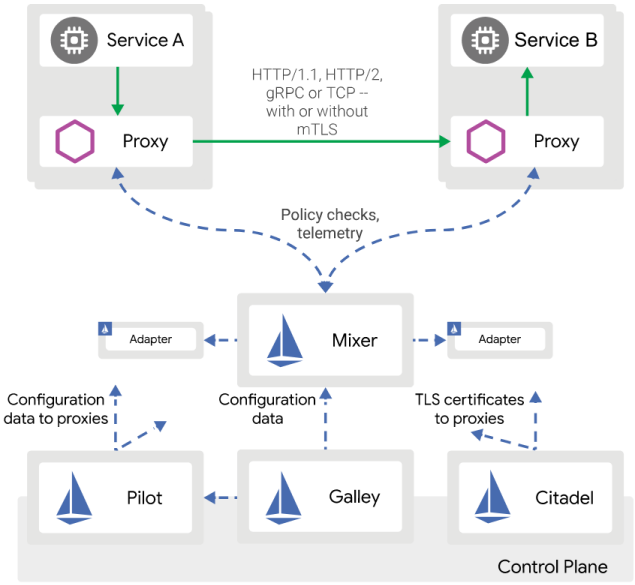

Figure 1: Istio Architecture

2. Data Plane
   (a) Envoy Proxies [3]: There is a network of these proxies, with each proxy paired with a service container.

2.2 GDPR

European Union in 2016 introduced a set of articles (99 to be precise) to enforce a set of rules and regulations to move the the control of data from the companies back to the users who actually own it [5]. GDPR (General Data Protection Regulation) provides an expansive set of rules to guide the entire lifetime of the data, but for our context we would be focusing on how these regulations impact application design. GDPR broadly divides the requirements under two categories, 1) Rights of the Users 2) Responsibilities of the Application Providers

2.2.1 Rights of the User

GDPR tries to educate the users about their rights in the digital domain, by laying down articles that move the ownership of the data from the applications providers back to the users. In article 15 of GDPR gives users right to ask the application providers about what they are storing, where and how long the data would be stored and for what purpose. Furthermore, article 18 gives users right to object to any of the data processing done by the providers anytime for any purpose. And in extreme cases, GDPR gives users right to ask the application providers to delete all the data associated with the user as specified in article 17. Lastly in article 20, it gives users the right

No.	GDPR Article	Key Requirement	Service Mesh Feature
5	Purpose Limitation	Data should be acquired for a specific reason	-
13	User Consent	Data can not be collected or processed without consent	Traffic Engineering Rules
15	Right of access by the user	Provide users access to their data	-
17	Right to be forgotten	Users data should be deleted on request	-
18	Right to restriction of processing	Users data should only be processed by consented services	Traffic Engineering Rules
20	Right to data portability	Users data should be transferred to other domains when requested	-
21	Right to object	Users can opt out of any previous consented service at any time	Traffic Engineering Rules
25	Data protection by design and default	Safeguard + Restrict access to data	mTLS
30	Records of Processing Activities	Logs of all operations	Logging
32	Security of processing	Implement state of the art security	mTLS, Access Policies
33/34	Data Breaches	Notify users about data breaches without undue delay	Auditing
46	Transfers subject to appropriate safeguards	Control where data is stored	Traffic Engineering Rules

Table 1: **Augmented Architecture Designs**

to port their data from one application provider to another in case they want to use some competitor's application.

2.2.2 Responsibilities of the Application Providers

In GDPR's article 24, it states that ultimate responsibility of all the data lies in the hands of the application providers. So its paramount for them to design their applications, under article 32, such that they guarantee security of user's data in presence of adversaries. Its responsibility of the application provider to make sure state of the art encryption and procedures are followed to protect user's data. In article 5 of GDPR states that data should be collected for a specific purposed and should not be used for anything else, its responsibility of the application provider to make sure that data is used responsibly and according to the consent given by the user. It is duty of the application provider, under article 30 and 46 respectively, to give the users logs of all the activities that have been performed with and on their data and to communicate to the users where the data is physically stored. Lastly, in case of a data breach, the application provider is obligated under article 33/34 to report to both users and authorities without undue delay.

The table 1 summarizes discussed articles.

3 Designing for Compliance

The fundamental goals of service meshes have led to the availability of a multitude of features which help in making applications GDPR compliant. In this section, we describe each design goal along with the features that come under its umbrella and how those features help in catering for respective GDPR articles.

3.1 Service Meshes Design Goals

Security: Service Meshes provide mTLS between each service (implemented by Envoy proxies (Section-2.1.1) and allow developers to define fine-grained access policies. As these security features are integrated in the proxies and the control plane, this allows the developers to automatically cater for *Article-25: Data protection by design and default* and *Article-32: Security of Processing* without modifying their application code-bases.

Traffic Management: As service meshes data planes operate on L7, one can add fine-grained policies i.e. at per user level. This allows developers to add per-user policies to ensure each users consent agreement is adhered to.

We envision that there would be automation policy generation methods in place rather than developers themselves specifying the policies. These automatic methods would allow developers not to worry about *Article-13: user consent*, *Article-18: Right to restriction of processing*, *Article-21: Right to object*, *Article-46: Transfers subjects to appropriate safeguards* as the automatically generated fine-grained policies would cater to them.

Observability: With each service being accompanied by a Layer 7 proxy, its becomes trivial to generate detailed telemetry for all the service communication in the application. This allows developer not to worry about *Article-34, 35: Data Breaches* and *Article-30: Records of processing activities*.

Also, with detailed telemetry + logging capabilities, developers also gain help for *Article-15: Right of access by the user* and *Article-20: Right to data portability* but they would need a GDPR compliant storage system to fully support these articles.

Other than the various articles mentioned in this section, there are other articles (*Article-5: Purpose Limitation*, *Article-15: Right of access by the user*, *Article-17: Right to be forgotten*, *Article-20: Right to data portability*), which although the service mesh can't directly cater for, given a GDPR compliant storage system [13, 14], service meshes can help in easing the burden on the developer. For example, in *Article-15: Right of access by the user*, assuming a GDPR compliant storage system, an API could exist, which when the user requests their data would be activated and would generate appropriate

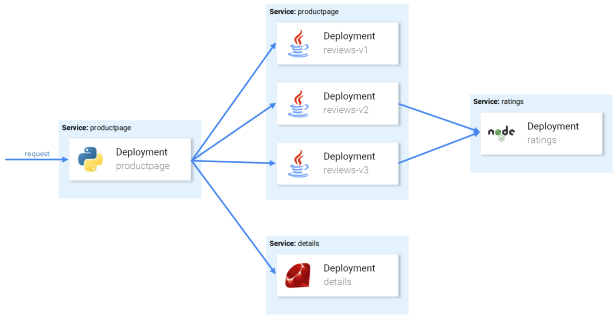

Figure 2: Bookinfo Application

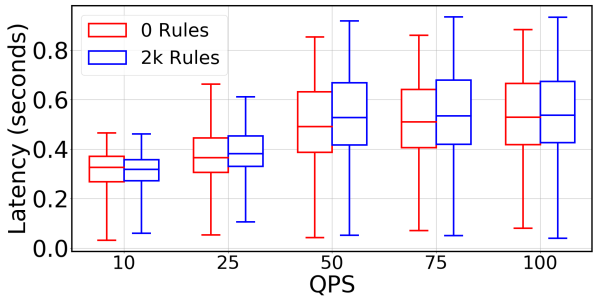

Figure 3: Policy Overhead

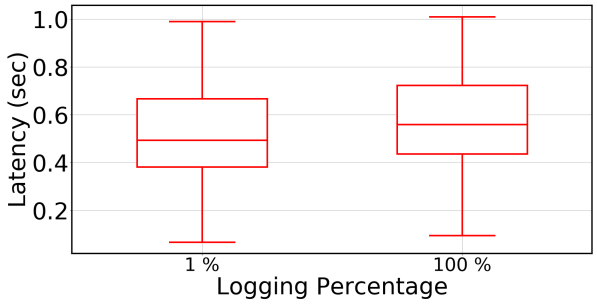

Figure 4: Logging Overhead

requests to the storage system and provide the user with their data.

4 Implementation

To evaluate the impact of applying user specific policies in a service mesh based application we decided to choose the Bookinfo application developed by Istio developers [1]. We deployed this sample application on Google cloud running under Kubernetes engine enabled instance.

To start off, the first thing that need to be implemented was a code that generated an arbitrary number of users with some randomized policies. This script also would get the users logged on with the website so that its cookie can be used later to run benchmarks. To run the benchmarks we used fortio [11] an application specifically designed to test throughput of the servers. We sent variable number of Queries per Second (ranging from 5 QPS to 100 QPS) with 25 threads (each thread sends the configured number of QPS) and for 120 seconds. This tool reports us the mean, median, min, max and specified quartile values of the response time distribution. For our analysis we plotted the trend in a box-whisker plot using 99th percentile as the max. To establish the impact of having user specific traffic engineering rules, we measured the server's throughput with and without policy rules. Results are discussed further in the evaluation section.

Similarly, we measured the impact of turning the logging on at different rates and measured how the server's throughput varied with that. To achieve that Istio provides a neat API to configure the rate of logging requests, we varied it from default 1 percent all the way up to 100 percent [12].

5 Evaluation

In this section, our goal was to evaluate how service mesh based applications performance degrades as various features are put in use to make the application GDPR compliant. For our evaluation, we leveraged Bookinfo [1] a small application shown in Figure-2.

In our experimentation we mainly focused on two features: (i) traffic engineering rules and (ii) logging. We wanted to quantify the performance degradation both these features add. To evaluate the performance degradation for these features, we leveraged a load generator: Fortio [11] designed by Google.

Following, we'll explain the experiments and results we observed.

5.1 Traffic Engineering Rules

In BookInfo [1], there are three versions of the **reviews** micro service. In this experiment, we added per-user rules where each user would only be directed to one of the three versions. We tested the application for 0 (no per-user rules added) and 2k users (2k rules added). Our results are presented in Figure-3.

Surprisingly, we saw very little performance degradation, even after we saturated (> 50 QPS the application. On closer inspection we realized that Istio caches the policy results in the proxies itself which leads to very little performance degradation. Moreover, we used our load generator against the base-page (of the website), hence there were no cache misses.

5.2 Logging

In this experiment, we enabled the distributed tracing feature. This allows one to re-construct all the requests thereby

constructing how each users data flowed in the system. We varied the logging rate from 1% (default) to 100% (each request is logged) and plotted the results in Figure-4. The degradation is still very minimal and this is due to the minimalist nature of the application we used.

6 Discussion + Limitations

Even though the results look promising but the evaluation is still missing some key features which would make it more realistic. The application that was considered for evaluation (even though its fully functional application) is still a very simple compared to most applications that are both deployed service meshes. So one extension of the evaluation would be to run the same set of experiments on both larger and more diverse set of applications.

Since we used Fortio to stress test our application, which by definition is the worst case for the application, is still not a realistic workload. So testing it on realistic workload traces would give a rather more realistic picture of how putting these policies and changes will impact performance.

Throughout our evaluation, we assumed that application is built on top of a GDPR compliant storage system. Implementing it from ground up would require non-trivial changes in storage system and would adversely impact the performance of the application as a whole.

7 Conclusion

In this paper, we've shown how service meshes already contain a multitude of features that make making applications GDPR compliant extremely easy. And our initial results show that performance degradation is minimal. Hence, this is a promising sign for developers to make their applications GDPR compliant with the help of service meshes.

References

[1] Bookinfo application.

[2] ebay data breach.

[3] Envoy proxy - home.

[4] Facebook gdpr compliance.

[5] Gdpr articles.

[6] Istio.

[7] Kuma.

[8] Linkerd.

[9] Marriott data breach.

[10] Yahoo data breach.

[11] FORTIO. fortio/fortio.

[12] READ, . M. Collecting logs.

[13] SCHWARZKOPF, M., KOHLER, E., FRANS KAASHOEK, M., AND MORRIS, R. Position: Gdpr compliance by construction. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare* (Cham, 2019), V. Gadepally, T. Mattson, M. Stonebraker, F. Wang, G. Luo, Y. Laing, and A. Dubovitskaya, Eds., Springer International Publishing, pp. 39–53.

[14] SHAH, A., BANAKAR, V., SHASTRI, S., WASSERMAN, M., AND CHIDAMBARAM, V. Analyzing the impact of {GDPR} on storage systems. In *11th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 19)* (2019).