****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

**Nathan Chuluda [CS-305-T1010 Software Security 21EW1](https://learn.snhu.edu/d2l/home/851359)**

**Table of Contents**

[Document Revision History 3](#_heading=h.gjdgxs)

[Client 3](#_heading=h.30j0zll)

[Instructions 3](#_heading=h.3znysh7)

[Developer 4](#_heading=h.2et92p0)

[1. Interpreting Client Needs 4](#_heading=h.tyjcwt)

[2. Areas of Security 4](#_heading=h.3dy6vkm)

[3. Manual Review 4](#_heading=h.1t3h5sf)

[4. Static Testing 4](#_heading=h.4d34og8)

[5. Mitigation Plan 4](#_heading=h.2s8eyo1)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9/13/2021** | **Nathan Chuluda** | **Initial report** |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Nathan Chuluda

## 1. Interpreting Client Needs

* **What is the value of secure communications to the company?**

As a financial institution, security is paramount in so many ways. The nature of the business requires processing, storing and retrieving sensitive information for your customers. Should customers lose trust in the safety of their finances and personal information, they will look to competitors who prove they can meet these needs. Around the world, there are also increasing legal obligations to protect customer data, such as the GDPR rules in Europe. In these countries there are defined responses to data breaches or other unauthorized access to this data. (*Data Protection and Online Privacy*, 2021)

* **Are there any international transactions that the company produces?**

Yes, it is almost guaranteed that a financial institution will have international transactions. Customers may be citizens of other or multiple countries. Investments can be made internationally as well. Customers may wish to transfer funds to accounts around the world.

* **Are there governmental restrictions about secure communications to consider?**

Absolutely. The EU General Data Protection Regulation rules are an example of obligations a business must take when collecting personal information from customers.

* **What external threats might be present now and in the immediate future?**

As soon as a system goes live, it is vulnerable for exploitation from attackers. A system should be designed with consideration of the existing known vulnerabilities in mind, as this is a good basis for developing a secure product. It must be assumed that the system will face a constant stream of malicious intent. In this sense, developers should strive to preemptively protect from threats that might arise in the future.

* **What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?**

Open source libraries can be a great resource when developing a system. It is common for software to share requirements and functionalities, and these open source libraries can save developers from always having to start from scratch. Instead, the collaboration of many users can be compounded into these projects, which can be shared by all. Security often benefits from the participation of a larger set of users, as there are more people to discover and correct vulnerabilities. That said, it is not safe to assume that all open source libraries will be inherently bug and risk free, so should open source libraries be used, care must still be taken to ensure that each meets the security needs of the organization.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

* **Input Validation**
  + This application accepts input from the user for /greeting?name=”*input*” in GreetingController.java. Currently, there is no restriction to how long this input can be, or what characters are permitted. Both extremely long input and special characters could pose potential risk for this input.
* **APIs**
  + This application uses a RESTful API in CrudController.java and GreetingController.java. As APIs accept input and respond with data, they are vulnerable points in the web application. Care should be taken to validate input and to check if the amount and type of data returned is reasonable.
* **Cryptography**
  + This application stores and accesses financial information for each customer. Personal information such as this should be encrypted as an additional measure to prevent unauthorized access during data transfer or from storage.
* **Client/Server**
  + This web application is an example of client / server architecture, as requests will be sent from the client and data will be returned from the server.
* **Code Error**
  + Any error messages displayed to the user should not disclose unnecessary details about the configurations of the system. Details such as these can guide users with malicious intent to refine their attacks so that they are tailored to the system. Errors should be logged so that developers can track bugs and monitor suspicious activity.
* **Code Quality**
  + Code should always follow best patterns and practices to ensure it is secure.
* **Encapsulation**
  + Encapsulation should be used to control how members of classes are accessed. This application has several classes, one of which contains sensitive information such as account number and account balance. These attributes should not be available to access by every component in the application.

## 

## 

## 

## 

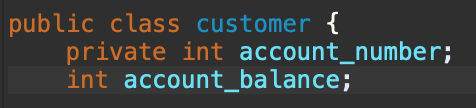
## 

## 

## 

## 3. Manual Review

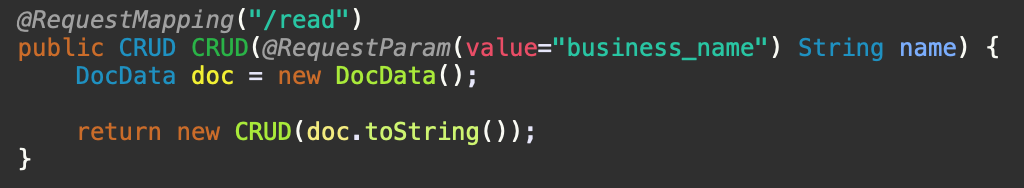
**customer.java**

****

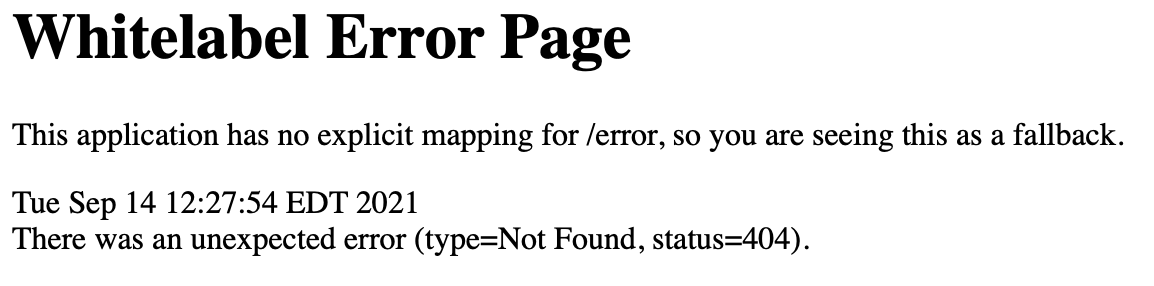
The field for account\_balance is not set to private. This is an example of not using encapsulation to secure the code. It is also an example of code quality not being up to par.

There is currently no encryption in use for these fields. Sensitive information such as account number and account balance should not be stored in cleartext.

**CRUDController.java**

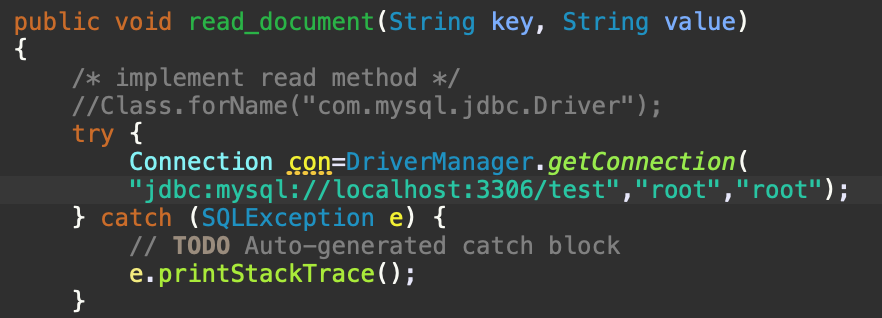
****

Input to this API parameter is not validated.



An error is produced when a user tries to access the /read endpoint defined in this file. The code should be corrected and completed, then reevaluated.

**DocData.java**

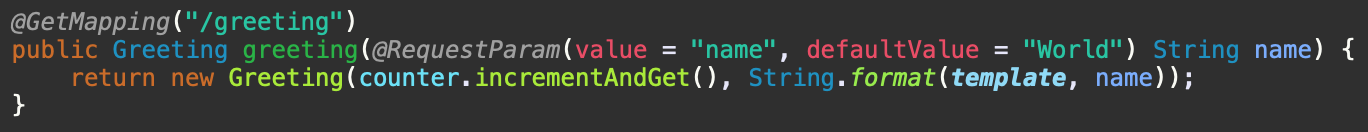
****

The username and password used for accessing the MySQL database is hardcoded in this file. Login credentials should never be hardcoded like this in the source code.

The read\_document() method is passed two parameters, but they are not currently utilized. This method should be reevaluated after it is completed.

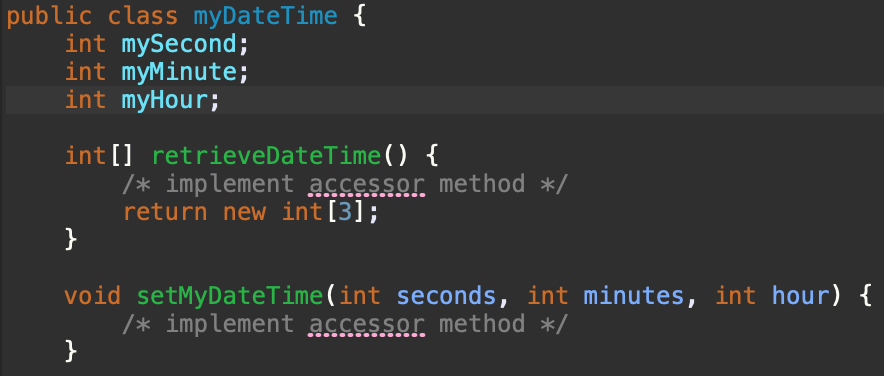
The method DocData() is defined in this class, but is not completed.

**GreetingController.java**

****

Input to this API parameter is not validated.

**myDateTime.java**



Encapsulation is not used for the members of this class.

The accessor method retrieveDateTime() just returns an empty array. Development on this method should be completed and reevaluated.

The setter method setMyDateTime() has not been completed. It should be finished and reevaluated.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

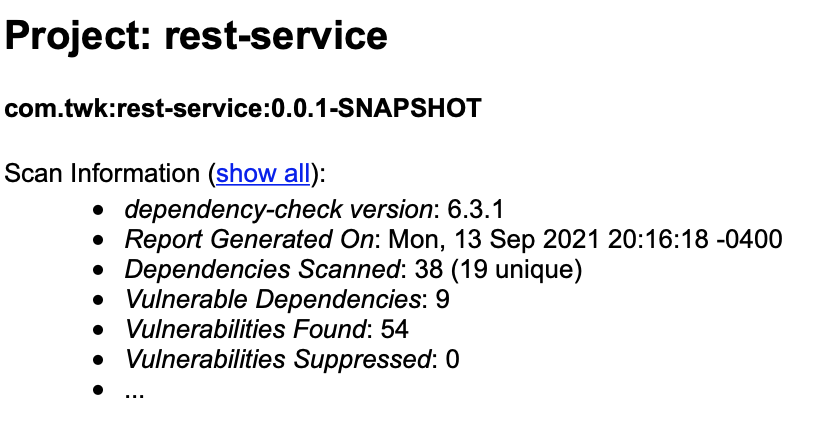
## 

## 

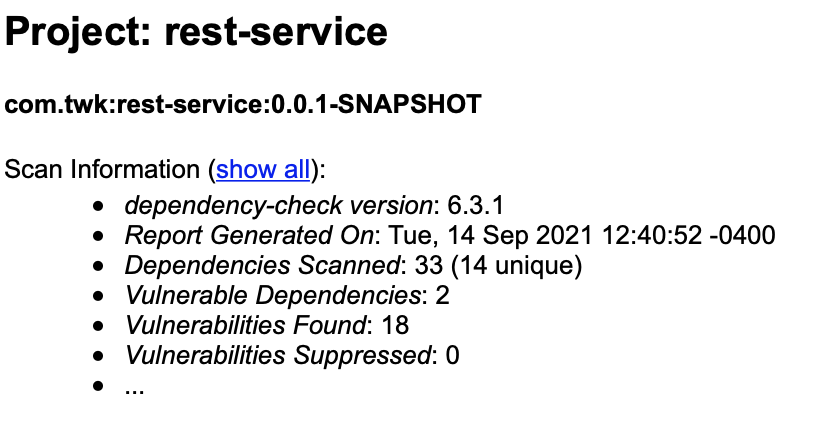
## 

## 4. Static Testing

The provided code does not utilize the current version of “spring-boot-starter-parent”, so initially the dependency check report has a significant amount of vulnerabilities listed.



Updating spring-boot-starter-parent from version 2.2.4 to 2.5.4 resolves many of these vulnerabilities.



There are now only two dependencies showing vulnerabilities in this dependency check report.

**The Bouncy Castle Crypto package**

**bcprov-jdk15on-1.46.jar**

| **CODE** | **DESCRIPTION / REMEDY** | **DOCUMENTATION** |
| --- | --- | --- |
| [**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624) | The TLS implementation in the Bouncy Castle Java library **before 1.48** and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169. | https://www.openwall.com/lists/oss-security/2013/02/05/24 |
| **CVE-2015-6644** | An information disclosure vulnerability in Bouncy Castle could enable a local malicious application to gain access to user?s private information | https://ossindex.sonatype.org/vulnerability/3a59870b-28b3-4b6b-86b0-9629ebe9de40?component-type=maven&component-name=org.bouncycastle.bcprov-jdk15on&utm\_source=dependency-check&utm\_medium=integration&utm\_content=6.3.1 |
| [**CVE-2015-7940**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940) | The Bouncy Castle Java library **before 1.51** does not validate a point is withing the elliptic curve, which makes it easier for remote attackers to obtain private keys via a series of crafted elliptic curve Diffie Hellman (ECDH) key exchanges, aka an "invalid curve attack." | https://www.openwall.com/lists/oss-security/2015/10/22/7 |
| [**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338) | In Bouncy Castle JCE Provider **version 1.55 and earlier** the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** the primary engine class used for AES was AESFastEngine. Due to the highly table driven approach used in the algorithm it turns out that if the data channel on the CPU can be monitored the lookup table accesses are sufficient to leak information on the AES key being used. There was also a leak in AESEngine although it was substantially less. AESEngine has been modified to remove any signs of leakage (testing carried out on Intel X86-64) and is now the primary AES class for the BC JCE provider from 1.56. Use of AESFastEngine is now only recommended where otherwise deemed appropriate. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** DSA signature generation is vulnerable to timing attack. Where timings can be closely observed for the generation of signatures, the lack of blinding in 1.55, or earlier, may allow an attacker to gain information about the signature's k value and ultimately the private value as well. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** ECDSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** the DSA key pair generator generates a weak private key if used with default values. If the JCA key pair generator is not explicitly initialised with DSA parameters, 1.55 and earlier generates a private value assuming a 1024 bit key size. In earlier releases this can be dealt with by explicitly passing parameters to the key pair generator. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000344**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** the DHIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider. | https://security.netapp.com/advisory/ntap-20181127-0004/ |
| [**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** the DHIES/ECIES CBC mode vulnerable to padding oracle attack. For BC 1.55 and older, in an environment where timings can be easily observed, it is possible with enough observations to identify when the decryption is failing due to padding. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** the other party DH public key is not fully validated. This can cause issues as invalid keys can be used to reveal details about the other party's private key where static Diffie-Hellman is in use. As of release 1.56 the key parameters are checked on agreement calculation. | https://lists.debian.org/debian-lts-announce/2018/07/msg00009.html |
| [**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352) | In the Bouncy Castle JCE Provider **version 1.55 and earlier** the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider. | https://security.netapp.com/advisory/ntap-20181127-0004/ |
| [**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098) | BouncyCastle TLS **prior to version 1.0.3**, when configured to use the JCE (Java Cryptography Extension) for cryptographic functions, provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application. This vulnerability is referred to as "ROBOT." | https://www.debian.org/security/2017/dsa-4072 |
| [**CVE-2018-1000613**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-1000613) | Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs **1.58 up to but not including 1.60** contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later. | https://security.netapp.com/advisory/ntap-20190204-0003/ |
| [**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382) | The default BKS keystore use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS keystore. Bouncy Castle release 1.47 changes the BKS format to a format which uses a 160 bit HMAC instead. This applies to any BKS keystore generated **prior to BC 1.47**. For situations where people need to create the files for legacy reasons a specific keystore type "BKS-V1" was introduced in 1.49. It should be noted that the use of "BKS-V1" is discouraged by the library authors and should only be used where it is otherwise safe to do so, as in where the use of a 16 bit checksum for the file integrity check is not going to cause a security issue in itself. | https://www.bouncycastle.org/releasenotes.html |
| [**CVE-2020-15522**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522) | Bouncy Castle BC Java **before 1.66**, BC C# .NET before 1.8.7, BC-FJA before 1.0.1.2, 1.0.2.1, and BC-FNA before 1.0.1.1 have a timing issue within the EC math library that can expose information about the private key when an attacker is able to observe timing information for the generation of multiple deterministic ECDSA signatures. | https://security.netapp.com/advisory/ntap-20210622-0007/ |
| **CVE-2020-26939** | In Legion of the Bouncy Castle BC **before 1.61** and BC-FJA before 1.0.1.2, attackers can obtain sensitive information about a private exponent because of Observable Differences in Behavior to Error Inputs. This occurs in org.bouncycastle.crypto.encodings.OAEPEncoding. Sending invalid ciphertext that decrypts to a short payload in the OAEP Decoder could result in the throwing of an early exception, potentially leaking some information about the private exponent of the RSA private key performing the encryption. | https://ossindex.sonatype.org/vulnerability/9e56f765-fe13-4d65-925a-241a8047f1b6?component-type=maven&component-name=org.bouncycastle.bcprov-jdk15on&utm\_source=dependency-check&utm\_medium=integration&utm\_content=6.3.1 |

**Core Tomcat implementation**

**tomcat-embed-core-9.0.52.jar**

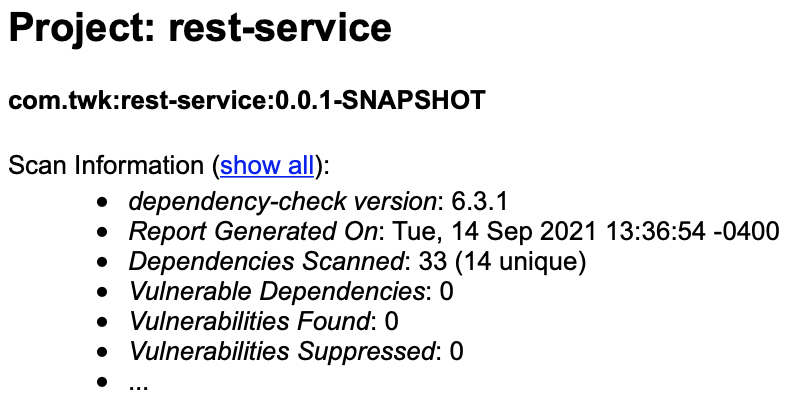
| **CODE** | **DESCRIPTION / REMEDY** | **DOCUMENTATION** |
| --- | --- | --- |
| [**CVE-2020-0822**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-0822) | An elevation of privilege vulnerability exists when the Windows Language Pack Installer improperly handles file operations, aka 'Windows Language Pack Installer Elevation of Privilege Vulnerability'. | https://msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-0822 |

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

**Action plan**

* Complete unfinished code, including methods and classes.
* Remove hardcoded login credentials for mySQL database from DocData.java.
* Use encapsulation for all classes.
* Implement encryption for sensitive data such as customer account number and account balance.
* Implement input validation for RESTful API endpoints /read and /greeting.
* Resolve error when accessing the /read endpoint.
* Update **bcprov-jdk15on-1.46.jar** and **tomcat-embed-core-9.0.52.jar** to the newest versions, **1.69** and **9.0.53** respectively. The following screenshot shows that this resolves the vulnerabilities found in the dependency check report.



After completing all steps of this action plan, a manual review of all code and static testing review should be completed again.

**Citations**

*Data protection and online privacy*. (2021, March 26). Your Europe. https://europa.eu/youreurope/citizens/consumers/internet-telecoms/data-protection-online-privacy/index\_en.htm