PROPOSAL FOR AN ONLINE GROCERY SHOPPING SYSTEM

Author: Anonymous for blind marking

Launching into Cyber Security

Dr. Maria Alvanou

University of Essex (Online)

October 9, 2023

Proposal for an online grocery shopping system

*I. Introduction*

Growing a physical store is challenging, due to physical limitations such as limited operating hours, customers based on geographical location, and high costs. An approach to grow without these limitations is to move business to an online shopping system. An online platform is always accessible to (new) customers and does not have the same high costs as a physical shop. Moving from a physical store to an online store does have its challenges, such as creating a safe platform. In the Netherlands, there are laws and regulations in place to ensure the safety of users as well as the owner. As part of the European Union, the General Data Protection Regulation (GDPR) should be followed by online stores in the Netherlands (*General Data Protection Regulation (GDPR) – Official Legal Text*, no date). This essay will propose a technical solution for a secure online shop that adheres to the GDPR, by using the Python Django framework.

*II. Needs for the system*

Before building any system, the requirements and needs for the system should be clear. As proposed by Sander Hoogendoorn (2004) the business processes of the online shopping system should be the starting point of the design. For the online grocery store, three key business processes can be identified as shown in Figure 1.

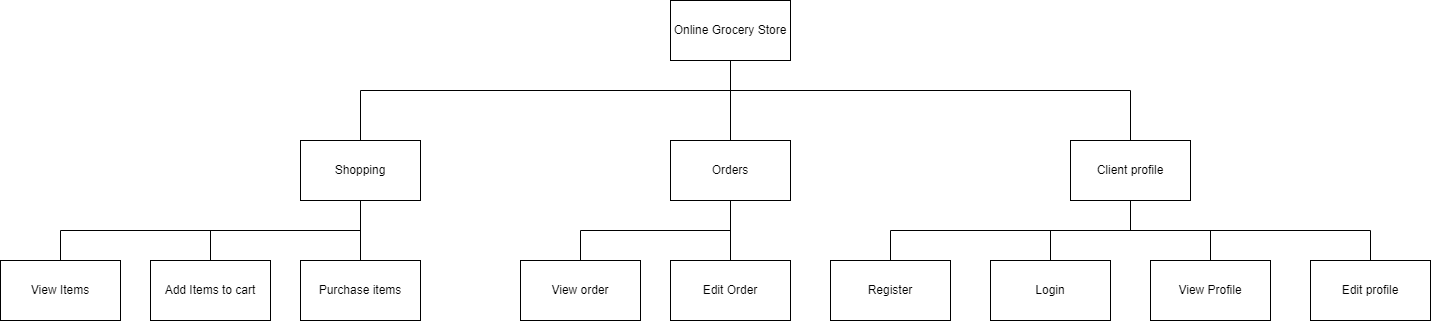


Figure 1 : Business proces diagram

Firstly, there is the “Shopping” process, enabling clients to purchase items. Next to that, the clients should be able to interact with their running and finished orders. Lastly, there should be room for a digital identity within the store. This digital identity will be used, so a customer can view his orders or shop via the online grocery store.

The “Client Profile” business process should be treated with care due to GDPR. Article 15 of the GDPR states that the data subject should have access to its data, resulting in the functionality of viewing orders and the data subject's profile (‘Art. 15 GDPR – Right of access by the data subject’, no date). Next to that article 17 of the GDPR states that the data subject has the right to be forgotten (‘Art. 17 GDPR – Right to erasure (“right to be forgotten”)’, no date).

Figures 2, 3, and 4 show the activity diagrams of business processes defined in Figure 1. All activity diagrams will briefly be described in the following sections. Figure 2 focuses on the activity diagram surrounding “Client processes”, this activity diagram already takes into account cyber security measures, by implementing a maximum of login attempts. Next to that, the edit profile activity diagram supports the functionality to delete a profile as part of Article 17 of the GDPR.

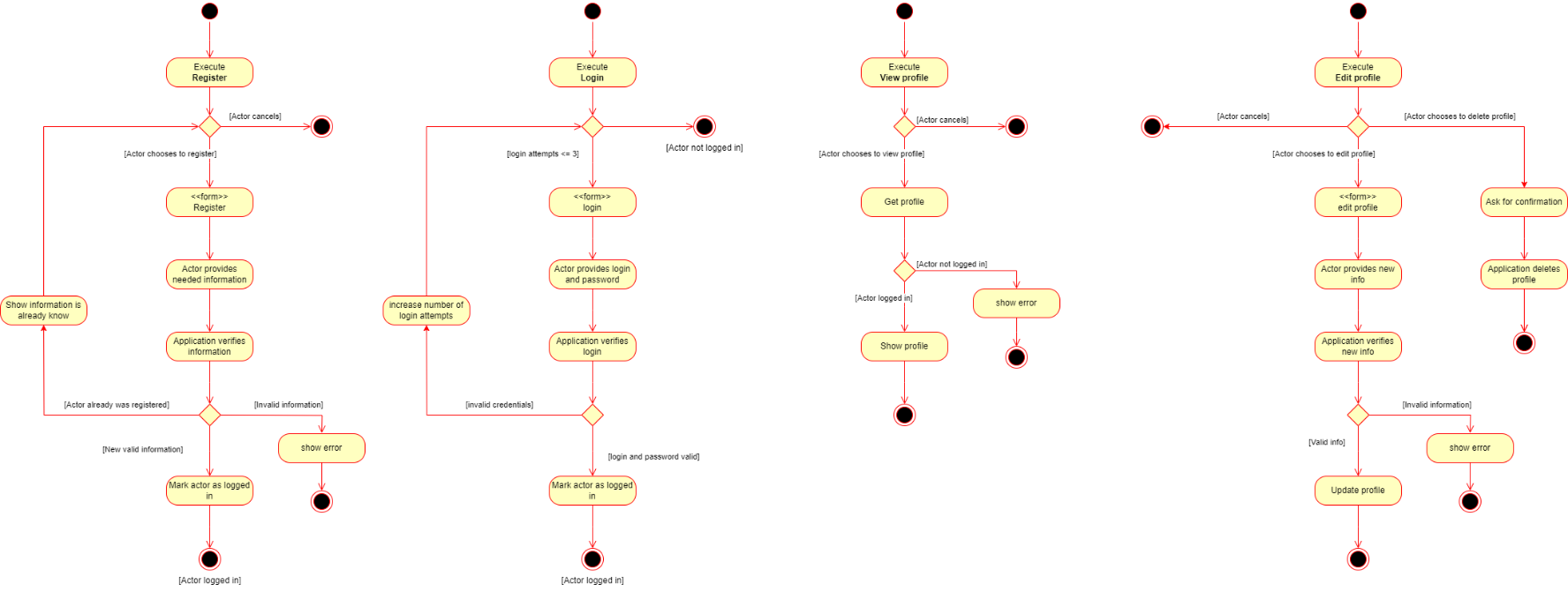


Figure 2: Client profile activity diagrams

Figure 3 models the activities of the business process of “Shopping”. On the right of the figure, there is a rudimentary activity diagram, showing the dependencies of the three activities “View items”, “Add items”, and “Purchase items”. Keep in mind that some activities rely on technical implementation, such as validating payment details, which would require third-party libraries.

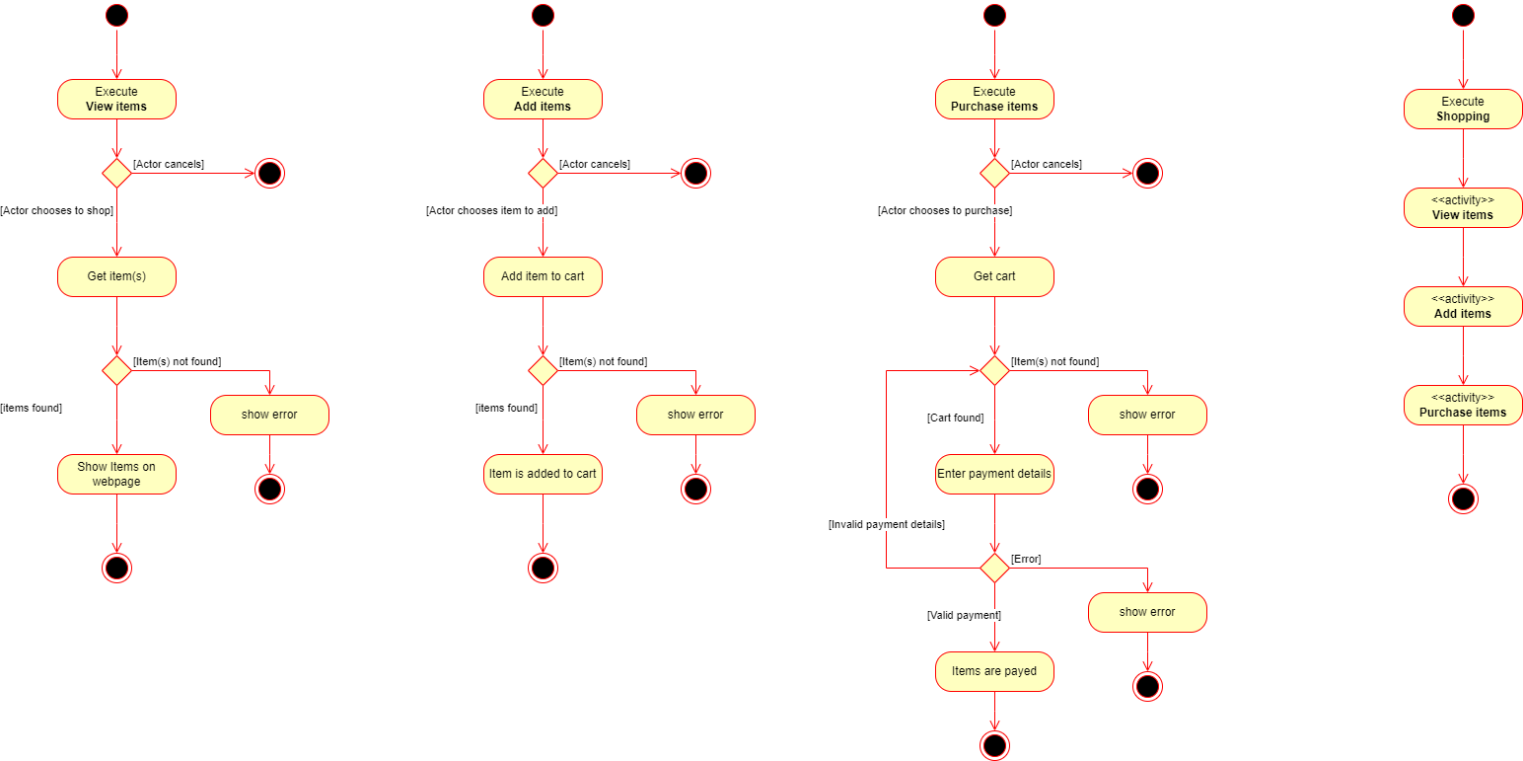
****

Figure 3: Activity diagrams shopping

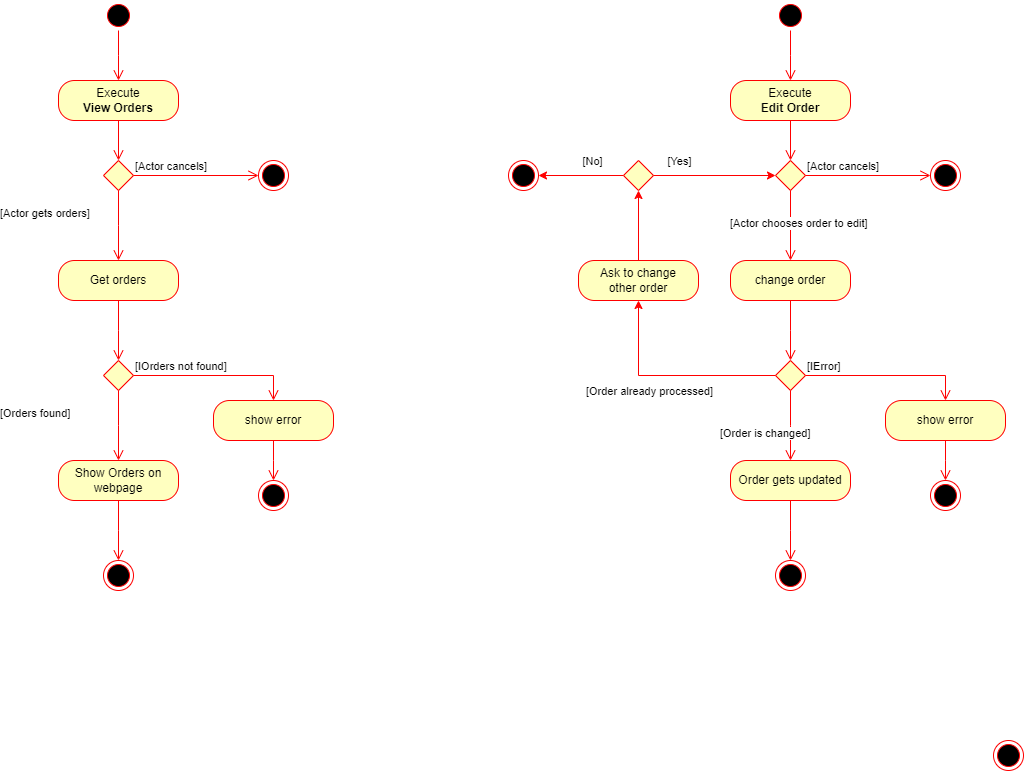
 Lastly, figure 4 models the activities of the business process “Orders”. Here Article 15 of the GDPR, as described previously, should be taken into account. The modeled activities give insight into the order data of the customer. Next to that, there is a possibility, as long as the order is not finished, to make additions to the order.

Figure 4: Activity diagrams Orders

Combining these processes and activities results in a fully functional online grocery store. In the technical design, some cyber security measures already have been implemented. More measures will be discussed in section IV.

*III. Proposal for a technical solution*

According to Soliev, Abdurasulova, and Yakubo (2022), the Python framework Django is well-known for building websites and is thus used by YouTube, Dropbox, and Spotify. Django is managed by the “Django Software Foundation”, which strives to protect the framework's long-term viability (*About the Django Software Foundation*, no date). This organization is dedicated to ensuring the safety of Django applications as it states the following: “Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery, and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.”(*Django overview*, no date)

Django uses the Model-View-Template (MVT) architecture to ensure the “Separation of Concerns”. Separation of Concerns is a commonly used principle in software development to create distinct sections in the application. The following paragraphs will look into the sections of MVT.

The model layer in MVT, the interface for the data, is responsible for defining the model classes in the application as well as mapping these classes to their database equivalent. This is done via an Object Relational Mapper (ORM) called SQLAlchemy. The database engine behind the application will be a MySQL database. The benefit of using an ORM vs “raw MySQL” is that an ORM enforces the application classes to match the database via migrations. SQLAlchemy still offers features to write “raw MySQL” statements instead of using the syntax of the ORM. One example based on the activities in Figure 3 would be that the Model layer is responsible for defining the features of an “Item” class.

The View layer in the MVT architecture encapsulates the logic responsible for processing a user’s request and for returning the response. In the case of this online shopping system, looking at Figure 3 the view layer would be responsible for handling the “Get Items” activity as well as returning the desired items on the web page.

Lastly, the MVT is made up of the Template layer, which defines the structure of the graphical interface for the users. One example could be an individual item on the items page, ideally, the data is consistently presented to the user, thus creating a template. To create a template Django uses the jinja2 syntax in combination with HTML, CSS, and JavaScript.

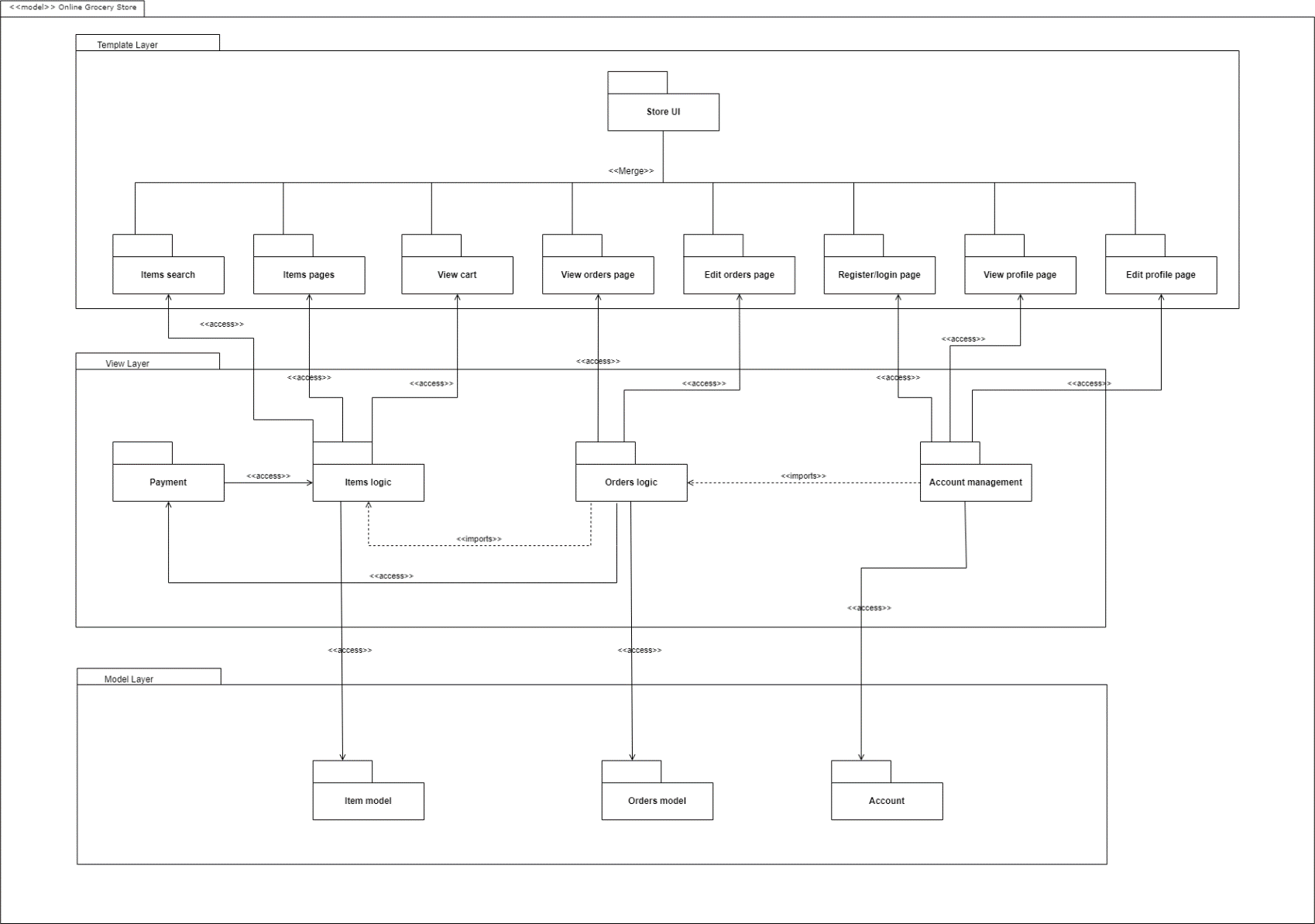
**By analyzing the activity diagrams of Figures 2,3, and 4 the following web pages can be derived: “Items search”, “Items pages”, “View cart”, “View orders page”, “Edit orders page”, “Register/login page”, “View Profile Page”, and “Edit Profile page”, all located in the Template layer. To support the logic needed for these pages the following functionalities are defined within the View layer: “Payment”, “Items logic”, “Orders logic”, and “Account management”. Lastly, the following model classes have been defined in the model layer: “Item model”, “Order model”, and “Account”. All these parts of the sections are intertwined between the different layers as shown by the package diagram in Figure 5.

Figure 5: Package Diagram

*IV. Ensuring the safety of the technical solution*

This section will focus on ensuring the cyber-resilience of the technical design of section III by applying threat modeling techniques. Some commonly used threat modeling techniques are STRIDE, attack trees, and the Common Vulnerability Scoring System (CVSS). STRIDE is an acronym of six different cyber threats: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. Attack trees begin by identifying a high-level threat and once that has been determined, a method is needed to decompose this threat into intermediate objectives of the attack (*Approaches to Security Design - Attack Tree and Protection Tree*, no date). CVSS is used to score threats based on multiple vectors, simplified it can be stated that it evaluates the risk/complexity against the potential damage, the higher the score the more critical a vulnerability is.

These three methods all have advantages and disadvantages. To get the best out of these three techniques the design will be evaluated via a Quantitative threat modeling method (QTMM). Effectively means that the Spoofing, Tampering, Denial of Service, and Elevation of privileges categories will be examined using attack trees together with the CVSS score. Lastly, mitigations for these threats will be proposed.

**Spoofing**

The spoofing category focuses on ensuring the authenticity of the user. The threat can be defined as pretending to be something or somebody other than yourself. Figure 6 defines the attack tree for the online grocery store in terms of spoofing.

A diagram of a system

Description automatically generated

Figure 6 : Spoofing attack tree

All paths leading to the user's credentials and info stolen have the same impact factors in their CVSS score. The CVSS vector string is: CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:L/A:N leading in a score of 5.9 medium. Django offers built-in security features against network eavesdropping, man-in-the-middle, and clickjacking (*Django*, no date). Django can also be configured to use HTTPS, encrypting data before it is transmitted over a network.

Brute forcing however has a different CVSS vector string: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N resulting in a score of 8.2 high. In the design of the activity diagram in Figure 2 the login attempts are already limited. Although this is a good first step in protecting against brute force attacks, another way to block brute force is a good password policy namely the NIST 800-63B series(*NIST Special Publication 800-63B*, no date).

**Tampering**

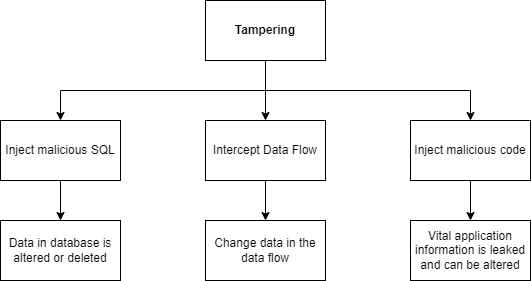
The tampering category is a violation of data integrity. This is the case when data is modified or altered. Figure 7 is a visualization of the attack tree for this category.

Figure 7 : Tampering attack tree

There are three possible vulnerable paths identified for tampering. SQL injection can result in altering or deleting the database and a complete loss of availability, confidentiality, and integrity. The CVSS vector string is: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H scoring a 10.0 critical. Django uses SQLAlchemy for queries, offering protection against SQL injection. However, Django supports raw SQL queries, these should be handled with care since there is no protection built in(*Django*, no date).

Data flow can be intercepted with the techniques discussed in the Spoofing section. The CVSS vector string is CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N resulting in a score of 7.5 high. As stated earlier built-in security components are protecting against these attacks.

Lastly, there is a possibility that somebody injects malicious code. There have been some known vulnerabilities in earlier versions of Django (*CVE-2023-31047 : In Django 3.2 before 3.2.19, 4.x before 4.1.9, and 4.2 before 4.2.1, it was possible to bypass validation when using one*, 2023). The CVSS vector string in this publication is: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H resulting in a score of 9.8 critical. To protect against this attack a safe version of Django must be chosen when realizing the application.

**Denial of Service**

The denial of service (DoS) category is aimed at the availability of the application and its resources. There has been a rise in Distributed Denial of Services (DDoS) (Mansfield-Devine, 2015), in addition to known application-level threats that are modeled in Figure 8.

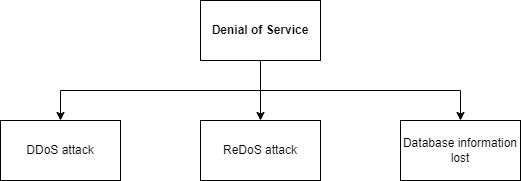


Figure 8: Denial of Service attack tree

The first vulnerable attack is a DDoS. The CVSS vector string is: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H with a score of 7.5 high. To protect against this attack the server of the application should implement a Web Application Firewall (WAF) (Gaylah, Vaghela, and Zongo, 2023).

Secondly, there is the ReDoS attack, where an evil regex is used to make the application “hang: (Owasp, 2019). The CVSS vector string is: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H resulting in a score of 7.5 high. The re module of Python can be used to mitigate this threat, allowing to escape malicious characters in such evil strings (*re — Regular expression operations*, no date).

Lastly, there is the deleted information threat. Multiple paths in other categories lead to this same threat, therefore no new CVSS score is determined. In case of this attack happening it is important to have a backup of the database. A rollback of this can result in a fast business continuation. A third-party library that exactly does this is “Django Database Backup” (*Django Database Backup — django-dbbackup 4.0.2 documentation*, no date), and thus will be implemented in the final application.

**Elevation of privilege**

The elevation of the privileges category is focused on gaining a higher level of authorization. Getting more authorization in an application means that a user gets rights they are not supposed to. Although the proposed application is flat in terms of authorization some “admin” features in the Django framework could lead to threats as modeled in Figure 9.

A diagram of a business flow

Description automatically generated

Figure 9: Elevation of Privilege attack tree

Getting admin rights means that an attacker has full access to the admin platform, resulting in control over database information. Meaning that an attacker can alter or delete all database information. The CVSS vector string is: CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H resulting in a score of 8.5 High. The DoS already discussed deleted information, which should be managed by using backups of the data.

*V. Conclusion*

This proposal aims to provide a technical solution for employing a secure online shop. Firstly, key business processes have been identified and modeled in activity diagrams. This activity diagram in combination with the MVT pattern used in Django applications has led to the technical design of the package diagram. A QTMM based on STRIDE, Attack trees and CVSS was done on this technical design. This approach resulted in **13** identifiedpotential threats. The following third-party libraries will be used to ensure the security: Django Database Backup, re, SQLAlchemy. Other potential threats were mitigated by the design or features built in Django. Lastly, a safe version of Django will result in a technical solution for a secure online shop that adheres to the GDPR, by using the Python Django framework.

Reference List

*About the Django Software Foundation* (no date) *Django Project*. Available at: https://www.djangoproject.com/foundation/ (Accessed: 9 October 2023).

*Approaches to Security Design - Attack Tree and Protection Tree* (no date). Available at: https://www.my-course.co.uk/Computing/Cyber%20Security/LCYS%202022/LCYS%202022%20Lecturecast%202/content/index.html#/lessons/IYY7CEvfPTJKc\_HI7a1k6SC573BgFGhE (Accessed: 9 October 2023).

‘Art. 15 GDPR – Right of access by the data subject’ (no date) *General Data Protection Regulation (GDPR)*. Available at: https://gdpr-info.eu/art-15-gdpr/ (Accessed: 9 October 2023).

‘Art. 17 GDPR – Right to erasure (“right to be forgotten”)’ (no date) *General Data Protection Regulation (GDPR)*. Available at: https://gdpr-info.eu/art-17-gdpr/ (Accessed: 9 October 2023).

*CVE-2023-31047 : In Django 3.2 before 3.2.19, 4.x before 4.1.9, and 4.2 before 4.2.1, it was possible to bypass validation when using one* (2023). Available at: https://www.cvedetails.com/cve/CVE-2023-31047/ (Accessed: 9 October 2023).

*Django* (no date) *Django Project*. Available at: https://docs.djangoproject.com/en/4.2/topics/security/ (Accessed: 9 October 2023).

*Django Database Backup — django-dbbackup 4.0.2 documentation* (no date). Available at: https://django-dbbackup.readthedocs.io/en/master/ (Accessed: 9 October 2023).

*Django overview* (no date) *Django Project*. Available at: https://www.djangoproject.com/start/overview/ (Accessed: 9 October 2023).

Gaylah, K.D., Vaghela, R.S. and Zongo, W.-B.S. (2023) ‘A Cost Optimized Solution for Defending Against DDoS Attacks: An Analysis of a Multi-layered Architecture’, *SN Computer Science*, 4(5), p. 631. Available at: https://doi.org/10.1007/s42979-023-02001-x.

*General Data Protection Regulation (GDPR) – Official Legal Text* (no date) *General Data Protection Regulation (GDPR)*. Available at: https://gdpr-info.eu/ (Accessed: 9 October 2023).

Hoogendoorn, S. (2004) *Pragmatisch modelleren met UML 2.0*. Pearson Education. Available at: https://books.google.nl/books?hl=en&lr=&id=6xZS5GL0d1MC&oi=fnd&pg=PR17&dq=pragmatisch+modelleren+met+UML&ots=8kb8tHZ1\_F&sig=x-qybfK0MdwcH4IBP8HPszGtcDE (Accessed: 9 October 2023).

Mansfield-Devine, S. (2015) ‘The growth and evolution of DDoS’, *Network Security*, 2015(10), pp. 13–20.

*NIST Special Publication 800-63B* (no date). Available at: https://pages.nist.gov/sp800-63b.html (Accessed: 9 October 2023).

Owasp, A. (2019) ‘Security project’, *OWASP Foundation, Inc.[cit. 2021-03-22]. Dostupné z: https://owasp. org/www-project-api-security* [Preprint].

*re — Regular expression operations* (no date) *Python documentation*. Available at: https://docs.python.org/3/library/re.html (Accessed: 9 October 2023).

Soliev, B.N., Abdurasulova, D. and Yakubov, M.S. (2022) ‘USING THE DJANGO FRAMEWORK FOR E-COMMERCE PROCESSES’, *Journal of Integrated Education and Research*, 1(6), pp. 229–233.