TripNau Threat Model

Owner: Group 2 - M1A
Reviewer:
Contributors: 1190326 - Afonso Machado, 1190535 - Domingos Machado, 1230201 - Nuno Ribeiro, 1230211 - Rui Neto, 1230212 - Simão Santos
Date Generated: Sun Apr 21 2024

OWASP Threat Dragon

Executive Summary

High level system description

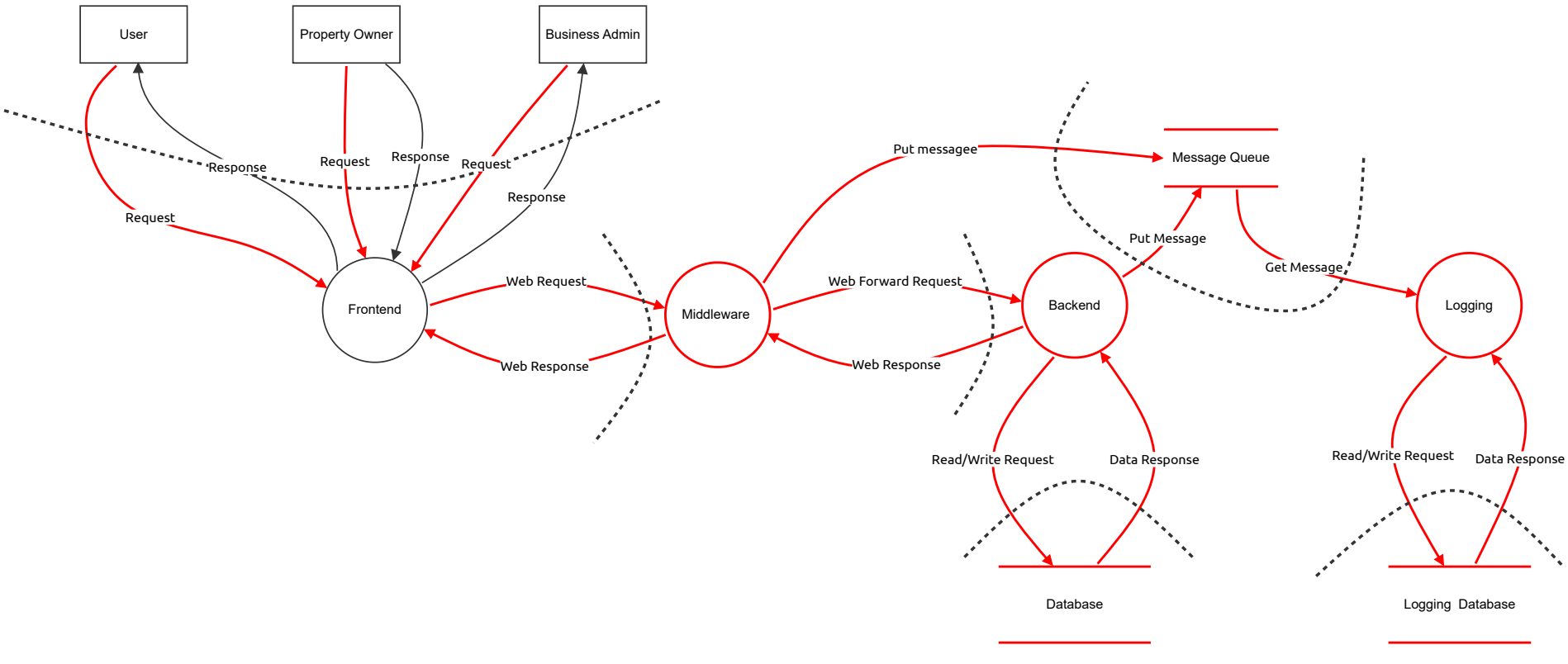
A Threat Model for the TripNau.

Summary

Total Threats	110
Total Mitigated	0
Not Mitigated	110
Open / High Priority	63
Open / Medium Priority	35
Open / Low Priority	12
Open / Unknown Priority	0

DFD - Level 1 - System High Level

The Higher Level Abstraction diagram



DFD - Level 1 - System High Level

Database (Store)

The database that store the generic data.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
	Unauthorized access	Information disclosure	High	Open		An attacker could make an query call on the DB.	Require all queries to be authenticated.

Request (Data Flow)

Generic Request

Number	Title	Type	Priority	Status	Score	Description	Mitigations
116	Accessing another user’s credentials	Spoofing	Medium	Open		The request could be done by an attacker that had access to the user's password through various means, such as phishing or intercepting login credentials, and because of that, the attacker can gain unauthorized access to the user's account, potentially compromising sensitive information, performing malicious actions, etc.	Two-Factor Authentication (2FA) adds an extra layer of security by requiring users to provide a second form of authentication, with a code sent to the mobile device, in addition to their password. Even if an attacker manages to steal the user's password, they would still need this second factor to access the account.

Request (Data Flow)

Generic Request

Number	Title	Type	Priority	Status	Score	Description	Mitigations
132	Accessing another user’s credentials	Spoofing	Medium	Open		The request could be done by an attacker that had access to the user's password through various means, such as phishing or intercepting login credentials, and because of that, the attacker can gain unauthorized access to the pending reviews page and approve or reject reviews.	Two-Factor Authentication (2FA) adds an extra layer of security by requiring users to provide a second form of authentication, with a code sent to the mobile device, in addition to their password. Even if an attacker manages to steal the user's password, they would still need this second factor to access the account.

Web Forward Request (Data Flow)

Generic web request.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
143	Data flow should use HTTP/S	Spoofing	Medium	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.
177	Can't identify the user that performs the operation	Repudiation	High	Open		There's a risk that requests will occur within the Middleware without proper attribution to the responsible user if there's no mechanism for attribution.	Implementing an audit log is a possible mitigation strategy. It records all significant events within the system, ensuring accountability and traceability.

Web Response (Data Flow)

Generic Response.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
142	Data flow should use HTTP/S	Information disclosure	Medium	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.
164	Man in the middle attack	Spoofing	Medium	Open		An attacker could intercept the request query in transit and change the content in order to block the receiver system.	Enforce an encrypted connection.

Data Response (Data Flow)

Query response.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
144	Credential Exposure	Information disclosure	Low	Open		If database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Read/Write Request (Data Flow)

Generic Query.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
127	Man in the middle attack	Information disclosure	Low	Open		An attacker could intercept the DB queries in transit and obtain sensitive information, such as DB credentials, query parameters or query results (is unlikely since the data flow is over a private network).	Enforce an encrypted connection at the DB server
136	SQL Injection	Tampering	High	Open		Attackers can manipulate the SQL auth queries by injecting malicious SQL code through input fields that interface with the database.	Use prepared statements and parameterized queries to handle SQL commands. Never concatenate user input directly into SQL queries. Regularly validate and sanitize all user inputs.

Request (Data Flow)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
133	Accessing another user’s credentials	Spoofing	Medium	Open		The request could be done by an attacker that had access to the user's password through various means, such as phishing or intercepting login credentials, and because of that, the attacker can gain unauthorized access to the pending reviews page and approve or reject reviews.	Two-Factor Authentication (2FA) adds an extra layer of security by requiring users to provide a second form of authentication, with a code sent to the mobile device, in addition to their password. Even if an attacker manages to steal the user's password, they would still need this second factor to access the account.

Read/Write Request (Data Flow)

The query to the database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
139	Log injection	Tampering	High	Open		Attackers can exploit vulnerabilities in the application's logging functionality corrupting the format of the file or injecting unexpected characters.	To mitigate log injection attacks, it should be implemented a proper input validation and sanitization techniques to ensure that user-supplied data cannot manipulate the log files.
166	Man in the middle attack	Spoofing	Low	Open		An attacker could intercept the DB queries in transit and obtain sensitive information, such as DB credentials, query parameters or query results (is unlikely since the data flow is over a private network).	Enforce an encrypted connection at the DB server

Data Response (Data Flow)

The response from the database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
145	Credential Exposure	Information disclosure	High	Open		If database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Put Message (Data Flow)

Send the message to the queue.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
148	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl)

Get Message (Data Flow)

Get the message.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
152	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Put message (Data Flow)

Send the message to the queue.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
147	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Web Request (Data Flow)

Endpoint of Middleware.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
141	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Web Response (Data Flow)

Generic response.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
140	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.
163	Man in the middle attack	Information disclosure	Medium	Open		An attacker could intercept the request query in transit and change the content in order to block the receiver system.	Enforce an encrypted connection.

Middleware (Process)

Responsible for request rate limiting, filtering, validation and sanitization before it reaches the backend.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
117	Middleware Interruption	Denial of service	High	Open		It's possible that the entry of the Middleware server can be target of a high demand requests by malicious actors with a high volume of traffic, overwhelming its resources and making it unable to respond to legitimate user requests. This results on the server of the application to becoming inaccessible or slow to respond for genuine users, disrupting its normal functioning.	One effective mitigation strategy is to implement a rate limiting and traffic filter mechanism, satisfying only the organic requests made by genuine users and blocking all suspicious interaction.
134	Missing identification of the operation's user	Repudiation	Medium	Open		There's a risk that requests will occur within the Middleware without proper attribution to the responsible user if there's no mechanism for attribution.	Implementing an audit log is a possible mitigation strategy. It records all significant events within the system, ensuring accountability and traceability.
149	Credentials exposure in repository	Information disclosure	Medium	Open		There's a risk of exposing credentials when source code is stored in a version control system and contains sensitive data such as usernames, passwords, or API keys.	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.

Backend (Process)

The backend application.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
123	Repudiation account from not validated accounts	Spoofing	Medium	Open		The backend server can receive request from users that have their account's detail fulfilment of random/invalid information. This can lead to a repudiation threat, because it will be impossible to detect the real person behind the user information.	A possible strategy to reduce the likelihood of this type of operation is to require all users to validate their email before gaining access to any functionality of the system.This type of measure can reduce the motivation for attackers to make anonymous requests. Note: A more secure and robust mitigation strategy is to implement Know Your Client (KYC), but given the scope of the application, this may be a barrier to new users entering the platform, so we take some risk to gain more users.
124	Poison messages	Denial of service	High	Open		An attacker could generate a malicious message that the Backend cannot process.	Validate the content of all messages, before processing. Reject any message that have invalid content and log the rejection. Do not log the malicious content - instead log a description of the error.
135	Can't identify the user that performs the operation	Repudiation	Medium	Open		There's a risk that requests will occur within the Backend without proper attribution to the responsible user if there's no mechanism for attribution.	Implementing an audit log is a possible mitigation strategy. It records all significant events within the system, ensuring accountability and traceability.
150	Credentials exposure in repository	Spoofing	Medium	Open		There's a risk of exposing credentials when source code is stored in a version control system and contains sensitive data such as usernames, passwords, or API keys.	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.
176	Cross-Site Scripting (XSS) attacks	Spoofing	High	Open		Cross-Site Scripting (XSS) poses a significant risk, allowing attackers to inject malicious scripts into web pages viewed by other users. By exploiting vulnerabilities in input fields or parameters, attackers can execute arbitrary code within the context of victims' browsers.	To mitigate the risk of XSS attacks, we can implement secure coding practices and input validation mechanisms, sanitizing and validating user inputs, especially those that are rendered in web pages.
179	Must attend to principle of least privilege on DB access	Tampering	High	Open		The component could be vulnerable to a tampering attack, where the attacker could maliciously modify some content in the database, such as deleting tables or changing profile database privileges.	One possible mitigation is to apply the principle of least privilege to the database used by this component. By granting only read/write access, we can minimize the damage provoked by this possible attack, where the attacker will be blocked from any admin operation.

Logging (Process)

The logging application.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
126	Admin credentials not shared over systems	Information disclosure	High	Open		With shared administrative credentials with the web server, if the web server credentials are compromised, the logging system is also compromised.	One mitigation strategy is to guarantee that exists a specialized logging admin credentials to access the logging system, instead of using the same that is used by web server admin.

Logging Database (Store)

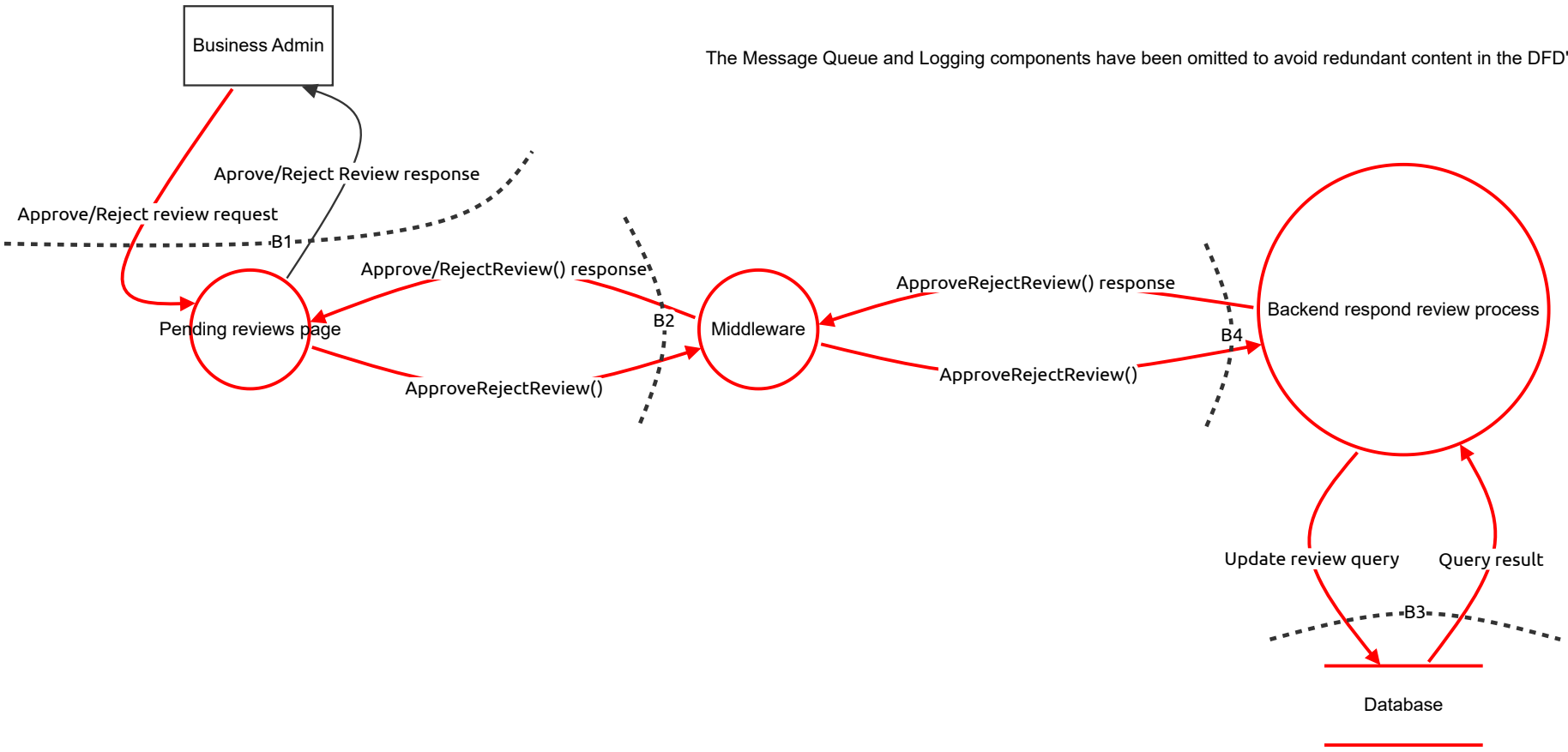
Number	Title	Type	Priority	Status	Score	Description	Mitigations
138	Unauthorized access	Information disclosure	High	Open		An attacker could make an query call on the application DB.	Require all queries to be authenticated.

Message Queue (Store)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
125	Message secrecy	Spoofing	Low	Open		The data flow between the Backend/Middleware and the Logging System is not point-to-point and therefore end-to-end secrecy cannot be provided at the transport layer. Messages could be read by an attacker at rest in the Message Queue.	Encrypt messages before enqueuing.
146	Fake messages could be placed on the queue	Spoofing	Medium	Open		An attacker could put a fake message on queue, causing the Background Worker to do incorrect processing.	Restrict access to the queue to the IP addresses of the Web Server and Background Worker. Implement authentication on the queue endpoint.

DFD - Level 2 - Approve/Reject Review Function

Approve/Reject Review Function diagram



DFD - Level 2 - Approve/Reject Review Function

Pending reviews page (Process)

Pending reviews page in the application frontend.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
28	Physically Insecure System	Elevation of privilege	Medium	Open		An attacker can physically access the pending reviews list and manipulate them.	I mitigation can be the re-login when accessing the pending reviews list. Another possibility would be the re-login on each approval/rejection. But that would take the user to an over consuming time.

Approve/Reject review request (Data Flow)

Approve/Reject review request data flow.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
9	Accessing another user's credentials	Spoofing	Medium	Open		The request could be done by an attacker that had access to the user's password through various means, such as phishing or intercepting login credentials, and because of that, the attacker can gain unauthorized access to the pending reviews page and approve or reject reviews.	Two-Factor Authentication (2FA) adds an extra layer of security by requiring users to provide a second form of authentication, with a code sent to the mobile device, in addition to their password. Even if an attacker manages to steal the user's password, they would still need this second factor to access the account.

ApproveRejectReview() (Data Flow)

Web request with the review ID to accept/reject.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
4	Data flow should use HTTP/S	Information disclosure	High	Open		These requests are made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentiality and integrity. HTTP should not be supported.

Approve/RejectReview() response (Data Flow)

Web response with the review updated status.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
3	Data flow should use HTTP/S	Information disclosure	High	Open		These requests are made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentiality and integrity. HTTP should not be supported.

ApproveRejectReview() (Data Flow)

Web request with the review ID to accept/reject.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
12	Data flow should use HTTP/S	Information disclosure	High	Open		These requests are made over the public internet and could be intercepted by an attacker.	These requests should require HTTP/S.This will provide confidentially and integrity. HTTP should not be supported.

ApproveRejectReview() response (Data Flow)

Web response with the review updated status.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
11	Data flow should use HTTP/S	Information disclosure	High	Open		These requests are made over the public internet and could be intercepted by an attacker.	These requests should require HTTP/S.This will provide confidentially and integrity. HTTP should not be supported.

Query result (Data Flow)

Update review query result.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
14	Man in the middle attack	Information disclosure	Medium	Open		An attacker could intercept the DB queries in transit and obtain sensitive information, such as DB credentials, query parameters or query results (is unlikely since the data flow is over a private network).	Enforce an encrypted connection at the DB server.
23	Credential Exposure	Information disclosure	Low	Open		If database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Update review query (Data Flow)

Query to update the requested review status to approved or rejected.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
19	SQL Injection	Tampering	High	Open		Attackers can manipulate the SQL auth queries by injecting malicious SQL code through input fields that interface with the database.	Use prepared statements and parameterized queries to handle SQL commands. Never concatenate user input directly into SQL queries. Regularly validate and sanitize all user inputs.
29	Must attend to principle of least privilege on DB access	Tampering	Medium	Open		The component could be vulnerable to a tampering attack, where the attacker could maliciously modify some content in the database, such as deleting tables or changing profile database privileges.	One possible mitigation is to apply the principle of least privilege to the database used by this component. By granting only read/write access, we can minimize the damage provoked by this possible attack, where the attacker will be blocked from any admin operation.
32	Man in the middle attack	Information disclosure	Low	Open		An attacker could intercept the DB queries in transit and obtain sensitive information, such as DB credentials, query parameters or query results (is unlikely since the data flow is over a private network).	Enforce an encrypted connection at the DB server

Middleware (Process)

The middleware responsible to sanitize and verify update review request authentication.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
24	Middleware Interruption	Denial of service	High	Open		Attacks such as Denial of Service (DoS) can overload the middleware with excessive requests, rendering it unavailable.	Implement rate limiting and proper load balancing.
33	Credentials exposure in repository	Information disclosure	High	Open		There's a risk of exposing credentials when source code contains sensitive information, such as usernames, passwords, or API keys, in a version control repository.	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.
193	File too large	Denial of service	Medium	Open		Provide a description for this threat	Provide remediation for this threat or a reason if status is N/A
194	File size too large	Denial of service	High	Open		Since there are endpoints that allow files to be sent, attackers are free to send files that are too large to cause a DOS when the file is processed or stored on the system.	To prevent this type of attack we need to set a file size limit and filename length limit.

Backend respond review process (Process)

The application backend that processes the review update request.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
27	Must attend to principle of least privilege on DB access	Tampering	High	Open		The component could be vulnerable to a tampering attack, where the attacker could maliciously modify some content in the database, such as deleting tables or changing profile database privileges.	One possible mitigation is to apply the principle of least privilege to the database used by this component. By granting only read/write access, we can minimize the damage provoked by this possible attack, where the attacker will be blocked from any admin operation.
30	Poison messages	Denial of service	High	Open		An attacker could generate a malicious message that the Backend cannot process.	Validate the content of all messages, before processing. Reject any message that have invalid content and log the rejection. Do not log the malicious content - instead log a description of the error.
34	Credentials exposure	Spoofing	High	Open		There's a risk of exposing credentials when code containing sensitive information, such as usernames, passwords, or API keys, is stored in a version control repository (repo).	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.
192	Malicious file upload	Denial of service	High	Open		Exists business logic on the Backend that often involves processing and storing files, creating a potential window for attackers to exploit vulnerabilities in the file parser. This could lead to scenarios where attackers overwrite other files or inject malicious client-side active content (XSS, CSRF, etc.).	To prevent this type of attack we need to: - Validate the file type, don't trust the Content-Type header as it can be spoofed - Change the filename to something generated by the application - Only allow authorized users to upload files

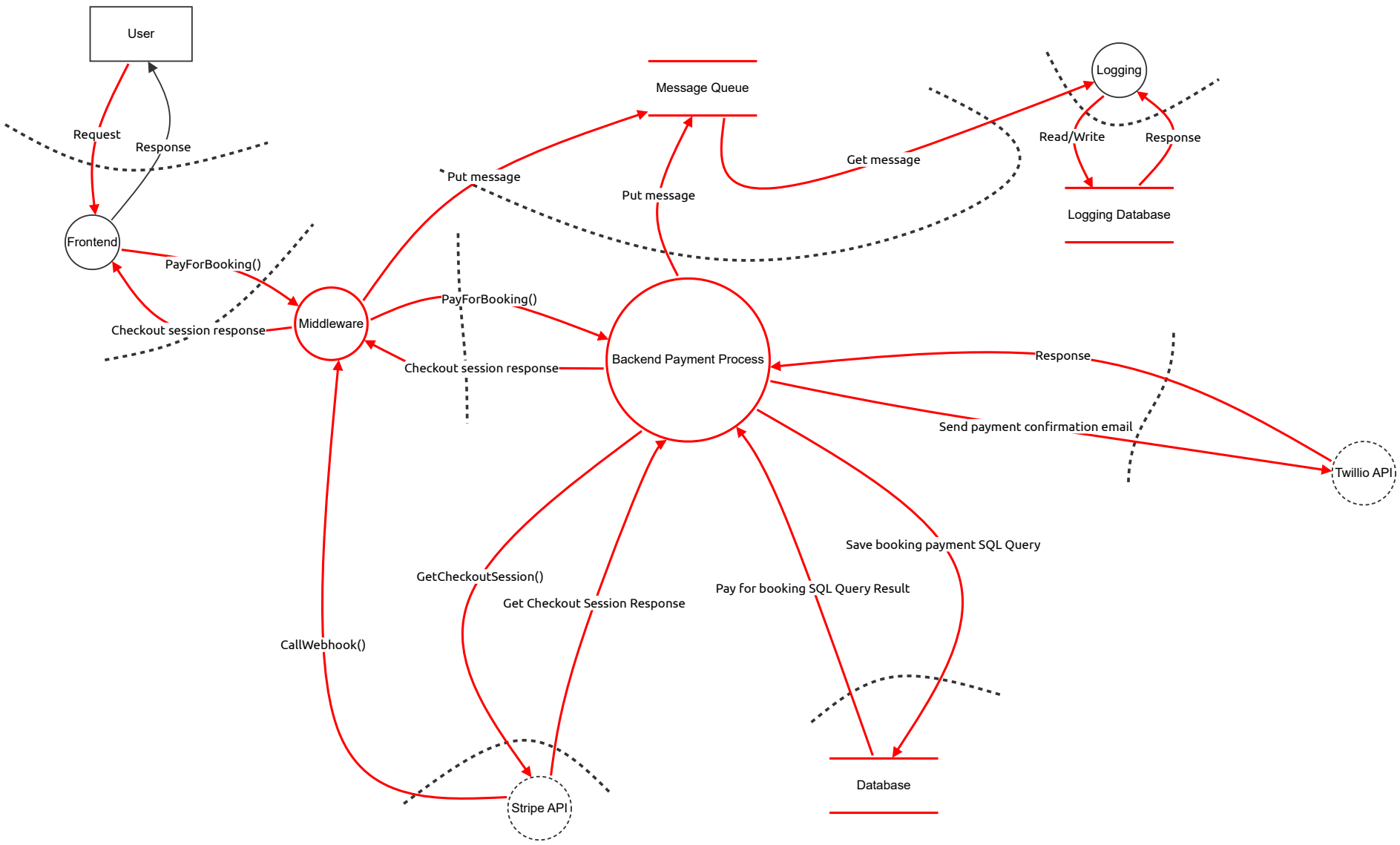
Database (Store)

Database that stores middleware auth tokens and the pending reviews data.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
15	Unauthorised access	Information disclosure	Medium	Open		An attacker could make an query call on the application DB.	Require all queries to be authenticated.

DFD - Level 2 - Booking Payment Function

Booking Payment Function diagram



DFD - Level 2 - Booking Payment Function

Request (Data Flow)

Click to pay for the booking.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
36	Accessing another user’s credentials	Spoofing	Medium	Open		The request could be done by an attacker that had access to the user's password through various means, such as phishing or intercepting login credentials, and because of that, the attacker can gain unauthorized access.	Two-Factor Authentication (2FA) adds an extra layer of security by requiring users to provide a second form of authentication, with a code sent to the mobile device, in addition to their password. Even if an attacker manages to steal the user's password, they would still need this second factor to access the account.

Checkout session response (Data Flow)

Stripe checkout session url.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
2	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

PayForBooking() (Data Flow)

Endpoint that provides the functionality related to the payment of bookings.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
5	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

CallWebhook() (Data Flow)

Webhook that the Stripe API executes whenever a payment is done.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
17	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.
18	Checkout Session metadata encryption	Information disclosure	High	Open		Third party service with access to domain relevant and private data.	GIven that a checkout session has linked metadata that is relevant to the application domain, this data must be encrypted so that the third party doesn't direct have access to it.

Get Checkout Session Response (Data Flow)

Stripe checkout session url.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
26	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.

GetCheckoutSession() (Data Flow)

Requests the Stripe API for a checkout session.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
19	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.
20	Checkout Session metadata encryption	Information disclosure	High	Open		Provide a description for this threat	Provide remediation for this threat or a reason if status is N/A

PayForBooking() (Data Flow)

Endpoint that provides the functionality related to the payment of bookings.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
6	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.
35	Can't identify the user that performs the operation	Repudiation	High	Open		There's a risk that requests will occur within the Middleware without proper attribution to the responsible user if there's no mechanism for attribution.	Implementing an audit log is a possible mitigation strategy. It records all significant events within the system, ensuring accountability and traceability.

Checkout session response (Data Flow)

Stripe checkout session url.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
9	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Pay for booking SQL Query Result (Data Flow)

SQL Query result.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
31	Credential Exposure	Information disclosure	Low	Open		if database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Save booking payment SQL Query (Data Flow)

Query that inserts payment data to the database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
28	SQL Injection	Tampering	High	Open		Attackers can manipulate the SQL queries by injecting malicious SQL code through input fields that interface with the database.	Use prepared statements and parameterized queries to handle SQL commands. Never concatenate user input directly into SQL queries. Regularly validate and sanitize all user inputs.
30	Credential Exposure	Information disclosure	Low	Open		If database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Response (Data Flow)

Response from the request.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
25	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.

Send payment confirmation email (Data Flow)

request to send an e-mail to a user with the purpose of notifying that a payment was done.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
11	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.

Put message (Data Flow)

Writes a new log at end of the queue.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
22	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Put message (Data Flow)

Writes a new log at end of the queue.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
23	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Get message (Data Flow)

Dequeue message and stores it on the logging database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
27	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Read/Write (Data Flow)

Reads ans writes data into the logging database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
32	Credentials exposure	Spoofing	High	Open		There's a risk of exposing credentials when code containing sensitive information, such as usernames, passwords, or API keys, is stored in a version control repository (repo).	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.

Response (Data Flow)

Request response

Number	Title	Type	Priority	Status	Score	Description	Mitigations
33	Credentials exposure	Spoofing	High	Open		There's a risk of exposing credentials when code containing sensitive information, such as usernames, passwords, or API keys, is stored in a version control repository (repo).	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.

Middleware (Process)

Responsible for request rate limiting, filtering, validation and sanitization before it reaches the backend.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
4	Middleware Interruption	Denial of service	High	Open		It's possible that the entry of the Middleware server can be target of a high demand requests by malicious actors with a high volume of traffic, overwhelming its resources and making it unable to respond to legitimate user requests. This results on the server of the application to becoming inaccessible or slow to respond for genuine users, disrupting its normal functioning.	One effective mitigation strategy is to implement a rate limiting and traffic filter mechanism, satisfying only the organic requests made by genuine users and blocking all suspicious interaction.
39	Credential Exposure in repository	Information disclosure	Medium	Open		There's a risk of exposing credentials when source code is stored in a version control system and contains sensitive data such as usernames, passwords, or API keys.	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.

Backend Payment Process (Process)

The application backend that orchestrates the logic of a booking payment.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
12	Poisoned messages	Denial of service	High	Open		An attacker could generate a malicious message that the Backend cannot process.	Validate the content of all messages, before processing. Reject any message that have invalid content and log the rejection. Do not log the malicious content - instead log a description of the error.
13	Must attend to principle of least privilege on DB access	Tampering	High	Open		The component could be vulnerable to a tampering attack, where the attacker could maliciously modify some content in the database, such as deleting tables or changing profile database privileges.	One possible mitigation is to apply the principle of least privilege to the database used by this component. By granting only read/write access, we can minimize the damage provoked by this possible attack, where the attacker will be blocked from any admin operation.
14	Repudiation account from not validated accounts	Spoofing	Medium	Open		The backend server can receive request from users that have their account's detail fulfilment of random/invalid information. This can lead to a repudiation threat, because it will be impossible to detect the real person behind the user information.	A possible strategy to reduce the likelihood of this type of operation is to require all users to validate their email before gaining access to any functionality of the system.This type of measure can reduce the motivation for attackers to make anonymous requests. Note: A more secure and robust mitigation strategy is to implement Know Your Client (KYC), but given the scope of the application, this may be a barrier to new users entering the platform, so we take some risk to gain more users.
40	Credential Exposure in repository	Information disclosure	Medium	Open		There's a risk of exposing credentials when source code is stored in a version control system and contains sensitive data such as usernames, passwords, or API keys.	To mitigate the risk of credential exposure, it's essential to avoid storing sensitive credentials directly in the codebase or version control repository. Instead, we should adopt secure credential management practices, such as using environment variables, configuration files outside of the repository, or a secure vault solution.

Database (Store)

Database that stores the application relevant data.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
29	Unauthorized access	Information disclosure	High	Open		An attacker could make an query call on the application DB.	Require all queries to be authenticated.

Message Queue (Store)

Queue of logs that have to be stored on a logging server.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
16	Fake messages could be placed on the queue	Spoofing	Medium	Open		An attacker could put a fake message on queue, causing the Background Worker to do incorrect processing.	Restrict access to the queue to the IP addresses of the Web Server and Background Worker. Implement authentication on the queue endpoint.
24	Message secrecy	Information disclosure	Low	Open		The data flow between the Web Application and the Background Worker is not point-to-point and therefore end-to-end secrecy cannot be provided at the transport layer. Messages could be read by an attacker at rest in the Message Queue.	Encrypt messages before enqueuing.

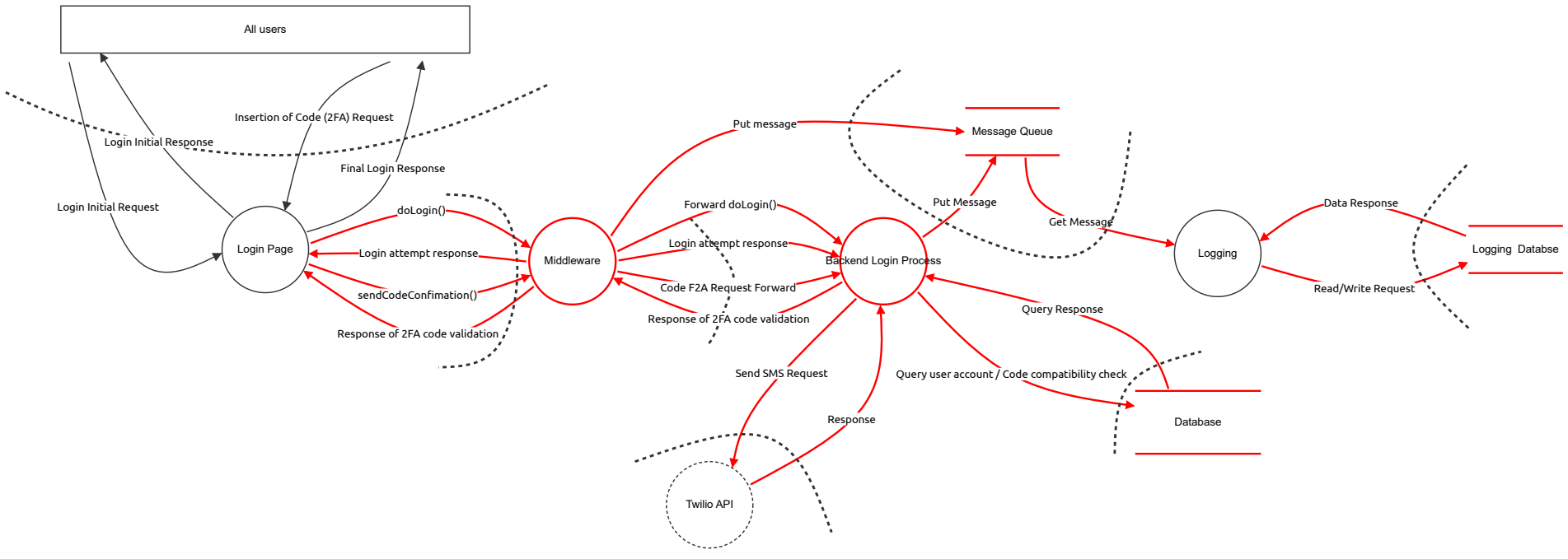
Logging Database (Store)

Database that stores the logs.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
34	Unauthorized access	Information disclosure	High	Open		An attacker could make an query call on the logs DB.	Require all queries to be authenticated.

DFD - Level 2 - Two Factor Authentication Function

Two Factor Authentication Function diagram



DFD - Level 2 - Two Factor Authentication Function

Database (Store)

Database that stores the application relevant data.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
	Unauthorized access	Information disclosure	High	Open		An attacker could make an query call on the DB.	Require all queries to be authenticated.

Login attempt response (Data Flow)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
143	Data flow should use HTTP/S	Spoofing	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Query Response (Data Flow)

SQL Query Result.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
144	Credential Exposure	Information disclosure	Low	Open		If database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Query user account / Code compatibility check (Data Flow)

Represents the query done to check the user account info or the 2FA code.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
127	Man in the middle attack	Information disclosure	Low	Open		An attacker could intercept the DB queries in transit and obtain sensitive information, such as DB credentials, query parameters or query results (is unlikely since the data flow is over a private network).	Enforce an encrypted connection at the DB server
136	SQL Injection	Tampering	High	Open		Attackers can manipulate the SQL auth queries by injecting malicious SQL code through input fields that interface with the database.	Use prepared statements and parameterized queries to handle SQL commands. Never concatenate user input directly into SQL queries. Regularly validate and sanitize all user inputs.

Read/Write Request (Data Flow)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
191	Credentials exposure	Tampering	Medium	Open		Provide a description for this threat	Provide remediation for this threat or a reason if status is N/A

Data Response (Data Flow)

Reads and writes data into the logging database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
145	Credential Exposure	Information disclosure	High	Open		If database credentials are transmitted insecurely, they could be intercepted and used by an attacker to gain unauthorized access.	Always use encrypted connections for transmitting credentials and employ robust authentication mechanisms.

Put Message (Data Flow)

Writes a new log at end of the queue.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
148	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl)

Get Message (Data Flow)

Dequeue message and stores it on the logging database.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
190	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Put message (Data Flow)

Writes a new log at end of the queue.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
147	Message should be encrypted	Information disclosure	High	Open		There's a risk of messages being intercepted by some attacker that can gain access to sensitive information with unauthorized access.	A mitigation strategy can be using encryption on the communication. This encryption should be end-to-end, meaning that only authorized parties possess the keys necessary to decrypt and access the contents of the messages. Using RabbitMQ we can use SSL (https://www.rabbitmq.com/docs/ssl).

Send SMS Request (Data Flow)

Endpoint to call on the external API to send the SMS with the 2FA code.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
188	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.

Response (Data Flow)

Response from the API.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
189	Compromised third party service	Tampering	Medium	Open		Risk of using a compromised third party service.	Having multiple alternatives provides the ability of deactivating the compromised API, reducing the risk of malicious actors tampering the email sending functionality or intercepting sensitive data transmitted through the compromised API.

Response of 2FA code validation (Data Flow)

The response that says if the user sent a valid or not valid 2FA code.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
185	Data flow should use HTTP/S	Information disclosure	Medium	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Response of 2FA code validation (Data Flow)

The response that says if the user sent a valid or not valid 2FA code.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
187	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

sendCodeConfimation() (Data Flow)

The endpoint that is called in order to confirm the 2FA code received by the user.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
184	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

doLogin() (Data Flow)

Call the login endpoint.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
140	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Forward doLogin() (Data Flow)

The forward request of the login

Number	Title	Type	Priority	Status	Score	Description	Mitigations
142	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Login attempt response (Data Flow)

The response of the first attempt login.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
141	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Code F2A Request Forward (Data Flow)

The endpoint that is called in order to confirm the 2FA code received by the user.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
186	Data flow should use HTTP/S	Information disclosure	High	Open		Given that the request is made over the public internet and could be intercepted by an attacker.	The requests should require HTTP/S. This will provide confidentially and integrity. HTTP should not be supported.

Middleware (Process)

Responsible for request rate limiting, filtering, validation and sanitization before it reaches the backend.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
117	Middleware Interruption	Denial of service	High	Open		It's possible that the entry of the Middleware server can be target of a high demand requests by malicious actors with a high volume of traffic, overwhelming its resources and making it unable to respond to legitimate user requests. This results on the server of the application to becoming inaccessible or slow to respond for genuine users, disrupting its normal functioning.	One effective mitigation strategy is to implement a rate limiting and traffic filter mechanism, satisfying only the organic requests made by genuine users and blocking all suspicious interaction.

Backend Login Process (Process)

The application backend that orchestrates the logic of login/2FA.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
123	Unknow origin from the senders	Repudiation	High	Open		The backend can receive brute force attacks for the login, and it can happen that we don't know where the origin is.	To mitigate this we can obtain the network information about the sender, e.g. like IP address.
124	Poison messages	Denial of service	High	Open		An attacker could generate a malicious message that the Backend cannot process.	Validate the content of all messages, before processing. Reject any message that have invalid content and log the rejection. Do not log the malicious content - instead log a description of the error.
155	Many retries failed	Spoofing	High	Open		There is an possibility of a brute force attack on the password or the code of 2FA.	A mitigation strategy can be block the victims account after 3 failed attacks, (3 for each operation), in order to preserve the integrity of the user's account.

Logging Databse (Store)

Database that stores the logs.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
138	Unauthorized access	Information disclosure	Medium	Open		An attacker could make an query call on the application DB.	Require all queries to be authenticated.

Message Queue (Store)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
125	Message secrecy	Information disclosure	Low	Open		The data flow between the Web Application and the Background Worker is not point-to-point and therefore end-to-end secrecy cannot be provided at the transport layer. Messages could be read by an attacker at rest in the Message Queue.	Encrypt messages before enqueuing.
146	Fake messages could be placed on the queue	Spoofing	Medium	Open		An attacker could put a fake message on queue, causing the Logging to do incorrect processing.	Restrict access to the queue to the IP addresses of the Middleware and Backend. Implement authentication on the queue endpoint.