New Threat Model

Owner: Jaytech enterprises
Reviewer:
Contributors:
Date Generated: Sat Nov 15 2025

Executive Summary

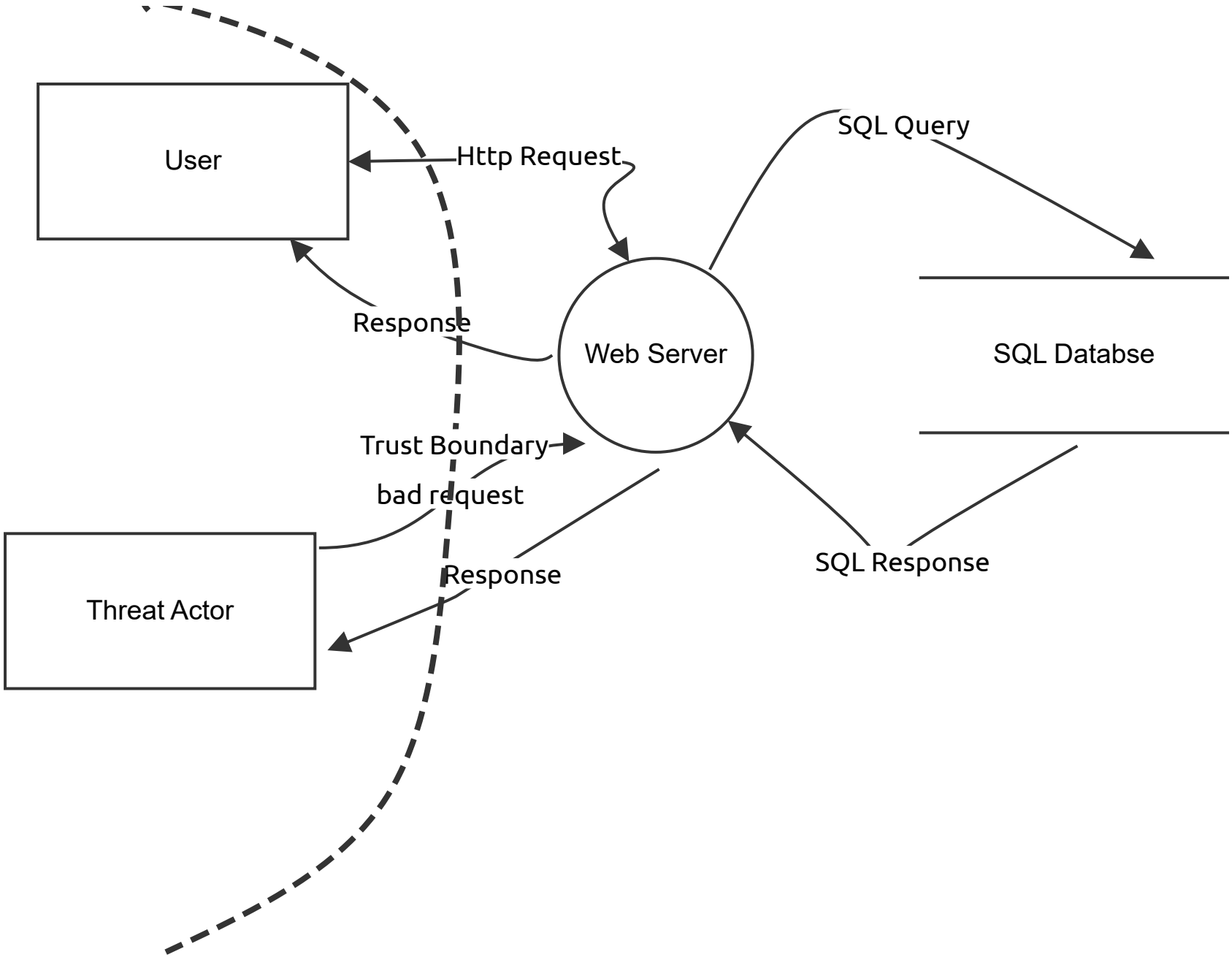
High level system description

Not provided

Summary

Total Threats	41
Total Mitigated	35
Total Open	6
Open / Critical Severity	0
Open / High Severity	0
Open / Medium Severity	6
Open / Low Severity	0

Main diagram DFD



Main diagram DFD

Web Server (Process)

Description: A web host

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Web Server Threat	Spoofing	Medium	Mitigated		Impersonating a legitimate user (session hijacking, stolen credentials) Spoofing the web server (DNS spoofing, fake site) Cookie/session token theft	Enforce HTTPS/TLS to prevent MITM Use strong authentication + MFA Use Secure, HttpOnly, SameSite cookies
	Web Server Threat	Tampering	Medium	Mitigated		Changing web files or server configuration Altering HTTP parameters/data Uploading malicious files (web shells) Manipulating data in transit if unencrypted	Use file integrity monitoring (Tripwire, AIDE) Validate & sanitize all inputs Store uploads outside the web root Apply proper file permissions (least privilege)
	Web Server Threat	Repudiation	Medium	Mitigated		No logs for user or admin actions Attackers deleting or modifying logs Lack of traceability for incidents	Enable full access & error logging Forward logs to a remote log server / SIEM Protect logs with strict permissions
	Web Server Threat	Information disclosure	Medium	Mitigated		Error messages exposing sensitive info Directory listing revealing server files Exposed configuration or backup files Unencrypted communication leaking credentials	Disable directory listing Mask server version banners (Apache/Nginx) Disable verbose error messages in production Use HTTPS/TLS everywhere
	Web Server Threat	Denial of service	Medium	Mitigated		Flooding server with excessive requests Slowloris/slow POST attacks Resource exhaustion (CPU, RAM, disk) Brute-force login attacks causing load	Use DDoS protection (Cloudflare, AWS Shield) Apply rate limiting & request throttling Enable connection timeouts Use WAF protections

User (Actor)

Description: A normal user that can be authenticated and authorised

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	User Threat	Spoofing	Medium	Mitigated		Someone logs in using the user’s stolen password Phishing attacks that trick the user into revealing credentials Fake login pages impersonating the real site Account takeover after session hijacking	Use strong, unique passwords Enable MFA (SMS, authenticator app, security key) Educate users about phishing Use password managers

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	User Threat	Tampering	Medium	Mitigated		Attackers altering stored user data (profile info, settings) Manipulation of user transactions (e.g., changing payment amount) Modified downloads (malicious software injected) Browser extensions altering page content	End-to-end encryption where possible Integrity checks on downloads (checksums, signatures) Use secure browsers and disable untrusted extensions Applications should validate all user actions before committing
	User Threat	Repudiation	Medium	Mitigated		User is accused of actions they didn't perform Lack of logs means user cannot dispute fraudulent changes Attackers perform actions that appear to come from user	Systems should maintain strong, timestamped logs Use email or SMS alerts for important account actions Allow users to review login history Use digital signatures for high-risk actions
	User Threat	Information disclosure	Medium	Mitigated		Personal data leaks (name, email, phone, DOB) Password leaks Credit card or financial info exposure Sensitive documents being accessed by attackers Tracking or surveillance without consent	Use strong encryption (HTTPS, encrypted storage) Avoid sharing sensitive data online unless necessary Enable privacy protections on devices and browsers Use secure password managers Regularly review data permissions for apps and websites
	User Threat	Denial of service	Medium	Mitigated		User is locked out of their account Online service becomes unavailable due to attack User's own device or browser becomes overloaded (e.g., malicious scripts)	Services should implement rate limiting and DDoS protection Users should keep browser and OS updated Avoid malicious sites and block ads/tracking scripts Use security software to prevent resource abuse

Threat Actor (Actor)

Description: A bad user that can be authenticated and authorised

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Http Request (Data Flow)

Description: Normal http request

Number	Title	Type	Severity	Status	Score	Description	Mitigations
104	HTTP threat	Tampering	TBD	Mitigated		Attacker modifies HTTP requests in transit to change parameters (e.g., price, quantity, or user ID) Attacker modifies cookies or session tokens Man-in-the-middle (MITM) attack changing request or response content Injection attacks via HTTP (e.g., modifying GET/POST payloads to exploit server)	Use HTTPS/TLS to encrypt HTTP traffic Validate and sanitize all incoming request parameters Sign or hash critical request payloads (digital signatures) Use secure, HttpOnly, SameSite cookies to prevent tampering Implement input validation and parameterized queries

Number	Title	Type	Severity	Status	Score	Description	Mitigations
105	Http threat	Information disclosure	TBD	Mitigated		Sensitive data sent in HTTP requests or responses without encryption (passwords, tokens, PII) Server error messages revealing system information Query strings in URLs exposing sensitive data Caching sensitive HTTP responses in client or proxy caches	Always use HTTPS/TLS to encrypt HTTP traffic Do not include sensitive information in URLs (use POST body or headers) Disable verbose error messages in production Set proper caching headers to prevent sensitive data storage in caches Apply security headers: Content Security Policy (CSP), X-Content-Type-Options, Strict-Transport-Security
106	Http threat	Denial of service	TBD	Mitigated		Flooding server with excessive HTTP requests (DDoS) Slow HTTP attacks (Slowloris) to exhaust server connections Sending malformed or extremely large HTTP requests to crash the server Repeated login attempts or API calls that consume resources	Rate limiting / throttling requests per IP or user Web Application Firewall (WAF) to detect & block DoS patterns Load balancing and autoscaling for capacity Limit maximum request size and timeout connections CAPTCHA or challenge-response for high-frequency endpoints

Response (Data Flow)

Description: Normal Response

Number	Title	Type	Severity	Status	Score	Description	Mitigations
118	web response to users threat	Tampering	TBD	Mitigated		An attacker modifies the HTTP response in transit (MITM attack). Malicious proxies inject scripts or content into the response (e.g., XSS injection). Altered headers or cookies to hijack sessions or bypass security	Use HTTPS/TLS to encrypt all responses. Apply Content Security Policy (CSP) and other security headers. Sign or hash critical response payloads if necessary. Validate server-side output before sending to the client.
119	Web Response to user threat	Information disclosure	TBD	Mitigated		Response leaks sensitive information (PII, tokens, internal server info). Verbose error messages or stack traces exposed to the user. Caching sensitive responses that can be read by others.	Remove sensitive data from responses. Disable verbose errors in production; show generic error messages. Use HTTPS/TLS to protect data in transit. Apply proper caching headers (Cache-Control: private, no-store). Filter response data according to user roles (least privilege).
120	Web Response to user threat	Denial of service	TBD	Mitigated		Extremely large responses slow down the user's browser or network. Malformed or malicious responses crash the client or cause resource exhaustion. Backend flooding responses that overwhelm user's connection.	Limit response size and implement pagination for large datasets. Optimize server-side processing to generate responses efficiently. Use rate limiting for high-volume endpoints. Apply network-level protections to prevent flooding.

bad request (Data Flow)

Description: Bad request from threat actor or bad users

Number	Title	Type	Severity	Status	Score	Description	Mitigations
111	Bad Request threat	Tampering	TBD	Mitigated		Request parameters are deliberately altered to bypass controls (e.g., changing user IDs, prices, or access flags). Malformed requests designed to exploit parsing bugs. Modifying headers, cookies, or query strings to manipulate server behavior.	Strict input validation and sanitation. Enforce parameter type and length checks. Use prepared statements for database queries. Reject malformed or unexpected HTTP requests.
112	Bad Request threat	Information disclosure	TBD	Mitigated		Malformed requests triggering verbose server errors revealing system info (stack traces, server versions, database names). Attackers probing endpoints and getting responses that leak internal details.	Disable verbose error messages in production. Mask server version banners and internal paths. Validate requests and reject invalid inputs gracefully. Use HTTPS/TLS to encrypt all traffic.
113	Bad Request threat	Denial of service	TBD	Mitigated		Sending repeated or malformed requests to overwhelm the server (DoS). Slow HTTP attacks (Slowloris) to exhaust connections. Extremely large requests to consume memory or CPU resources.	Rate limiting and throttling per IP/user. Connection timeouts and request size limits. Web Application Firewall (WAF) to detect suspicious patterns. Load balancing and autoscaling to handle traffic spikes.

Response (Data Flow)

Description: Response from the web server to bad users request

Number	Title	Type	Severity	Status	Score	Description	Mitigations
115	Server Response threat	Tampering	TBD	Mitigated		Response content modified in transit (MITM attack) Proxy or intermediary injecting malicious content (e.g., JavaScript) Attackers altering headers or cookies in the response to exploit clients	Use HTTPS/TLS for all responses Sign or hash critical response payloads if necessary Set secure HTTP headers (Content-Security-Policy, HSTS, X-Content-Type-Options) Validate server-side data before sending it to clients
116	Server Response threat	Information disclosure	TBD	Mitigated		Sensitive information included in responses (e.g., PII, tokens, passwords) Verbose error messages revealing internal server info (stack traces, database details) Response caching exposing private data to other users	Remove sensitive information from responses Disable verbose error messages in production Use HTTPS/TLS to encrypt responses in transit Set proper caching headers (Cache-Control, Pragma) Apply role-based filtering for returned data
117	Server Response threat	Denial of service	TBD	Mitigated		Large or complex responses cause client or network resource exhaustion Malformed response triggers client crashes Flooding responses (from backend) exhaust network bandwidth	Limit response size and implement pagination Optimize server-side queries and processing Rate limit high-volume endpoints Use load balancing / autoscaling

SQL Response (Data Flow)

Description: SQL response to web server request

Number	Title	Type	Severity	Status	Score	Description	Mitigations
107	SQL Response threat	Tampering	TBD	Mitigated		An attacker intercepts the SQL response and modifies it in transit (e.g., changing query results). Malicious middleware or proxy alters database responses before reaching the client. Corruption of database responses due to SQL injection exploits.	Use TLS/SSL to encrypt SQL traffic. Implement digital signatures or checksums on critical data. Limit privileges so that users cannot modify unauthorized data. Enable database auditing to detect unexpected changes.
108	SQL Response threat	Information disclosure	TBD	Mitigated		SQL responses leak sensitive data (PII, passwords, financial info) due to misconfigured queries or excessive permissions. Error messages in responses reveal database schema or internal implementation. Responses sent over unencrypted channels can be intercepted (MITM attack).	Encrypt SQL responses in transit (TLS/SSL). Apply least privilege for database queries and roles. Avoid exposing sensitive columns in responses. Disable verbose database error messages. Use parameterized queries and input validation to prevent SQL injection.
109	SQL Response threat	Denial of service	TBD	Mitigated		Large or complex queries return massive responses, overwhelming the client or network. Repeated queries exhaust database resources, causing slow or failed responses.	Implement query timeouts and result size limits. Use database connection pooling to prevent exhaustion. Monitor and rate-limit queries from users. Optimize queries and indexing to reduce heavy response generation.

SQL Query (Data Flow)

Description: web server query to data base

Number	Title	Type	Severity	Status	Score	Description	Mitigations
121	SQL query threat	Tampering	TBD	Mitigated		Query altered in transit (MITM attack) to modify database behavior. SQL injection exploits manipulate the query to access or modify unauthorized data. Malicious middleware modifies queries before reaching the database.	Use TLS/SSL to encrypt query traffic. Use parameterized queries / prepared statements to prevent SQL injection. Validate input before constructing queries. Monitor queries for abnormal patterns.
125	SQL query threat	Information disclosure	TBD	Mitigated		Queries returning sensitive information to unauthorized clients. Exploitation of poorly restricted queries exposing internal schema or PII. Logging queries without masking sensitive data exposes secrets.	Apply least privilege on database roles and queries. Encrypt sensitive data at rest and in transit. Avoid logging sensitive information. Filter query results based on user authorization.
126	SQL query threat	Denial of service	TBD	Mitigated		Maliciously crafted queries that consume excessive database resources (CPU, memory). Large or complex queries that block other legitimate requests. Repeated queries causing connection pool exhaustion.	Implement query timeouts and limits. Optimize database indexing and query performance. Monitor query patterns and apply rate limiting where possible. Use database connection pooling and limit maximum connections.

SQL Databse (Store)

Description: a data base that stored all the needful information

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	SQL Threat	Spoofing	Medium	Open		Threats involving impersonation or unauthorised identity use; Using stolen or guessed database credentials Bypassing authentication (e.g., via SQL injection that tricks login checks) Abuse of weak authentication methods (no MFA, weak passwords) Impersonating a database user through session hijacking	Enforce strong authentication (MFA, strong passwords, key-based auth) Use role-based access control (RBAC) Store credentials securely (Secrets manager: Vault, AWS Secrets Manager) Use TLS for DB connections to prevent session hijacking Implement account lockout + monitoring for brute force attempts Rotate passwords/service accounts regularly
	SQL Threat	Tampering	Medium	Open		Threats involving unauthorized modification of data: SQL injection used to modify database records Unauthorized updates, deletes, or schema changes Altering stored procedures, triggers, or configuration settings Manipulating data in transit if connections are not encrypted	Use parameterized queries / prepared statements Enforce least privilege (no app should run as DBA!) Enable DB integrity controls (checksums, row-level checks)
	SQL Threat	Repudiation	Medium	Open		Threats where actions can't be traced: Lack of proper audit logs for SQL activity Attackers clearing or modifying logs to hide activities	Enable comprehensive auditing (query logs, access logs, admin actions) Use append-only or external logging (SIEM: Splunk, ELK, CloudWatch) Protect logs with strict access controls
	SQL Threat	Information disclosure	Medium	Open		SQL injection revealing data Unencrypted connections or backups Overly broad SELECT permissions	Encrypt data in transit (TLS) Encrypt data at rest (TDE) Apply column-level encryption for sensitive fields Restrict SELECT access using RBAC and row-level security Avoid verbose database errors displayed to users Secure backups (encryption + restricted access)
	SQL Threat	Denial of service	Medium	Open		Heavy query floods Locking tables or long transactions Exhausted connection pool	Use rate limiting at the application/API layer Configure query timeouts Set maximum connection limits + pool management Monitor resource usage with alerts (CPU, disk, locks) Implement indexing and query optimization Enable automatic log rotation and disk quota monitoring
	SQL Threat	Elevation of privilege	Medium	Open		SQL injection executing privileged commands Vulnerable stored procedures running as admin Misconfigured roles or default accounts SQL engine vulnerabilities	Use least privilege everywhere Ensure stored procedures run under restricted execution context Disable or delete default database accounts Apply regular patches / updates to the DB server Use Web Application Firewalls (WAF) to block malicious SQL patterns Conduct periodic privilege audits

SQL Databse (Store)

Description: a data base that stored all the needful information

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	SQL Threat	Spoofing	Medium	Mitigated		Threats involving impersonation or unauthorised identity use; Using stolen or guessed database credentials Bypassing authentication (e.g., via SQL injection that tricks login checks) Abuse of weak authentication methods (no MFA, weak passwords) Impersonating a database user through session hijacking	Enforce strong authentication (MFA, strong passwords, key-based auth) Use role-based access control (RBAC) Store credentials securely (Secrets manager: Vault, AWS Secrets Manager) Use TLS for DB connections to prevent session hijacking Implement account lockout + monitoring for brute force attempts Rotate passwords/service accounts regularly
	SQL Threat	Tampering	Medium	Mitigated		Threats involving unauthorized modification of data: SQL injection used to modify database records Unauthorized updates, deletes, or schema changes Altering stored procedures, triggers, or configuration settings Manipulating data in transit if connections are not encrypted	Use parameterized queries / prepared statements Enforce least privilege (no app should run as DBA!) Enable DB integrity controls (checksums, row-level checks)
	SQL Threat	Repudiation	Medium	Mitigated		Threats where actions can't be traced: Lack of proper audit logs for SQL activity Attackers clearing or modifying logs to hide activities	Enable comprehensive auditing (query logs, access logs, admin actions) Use append-only or external logging (SIEM: Splunk, ELK, CloudWatch) Protect logs with strict access controls
	SQL Threat	Information disclosure	Medium	Mitigated		SQL injection revealing data Unencrypted connections or backups Overly broad SELECT permissions	Encrypt data in transit (TLS) Encrypt data at rest (TDE) Apply column-level encryption for sensitive fields Restrict SELECT access using RBAC and row-level security Avoid verbose database errors displayed to users Secure backups (encryption + restricted access)
	SQL Threat	Denial of service	Medium	Mitigated		Heavy query floods Locking tables or long transactions Exhausted connection pool	Use rate limiting at the application/API layer Configure query timeouts Set maximum connection limits + pool management Monitor resource usage with alerts (CPU, disk, locks) Implement indexing and query optimization Enable automatic log rotation and disk quota monitoring
	SQL Threat	Elevation of privilege	Medium	Mitigated		SQL injection executing privileged commands Vulnerable stored procedures running as admin Misconfigured roles or default accounts SQL engine vulnerabilities	Use least privilege everywhere Ensure stored procedures run under restricted execution context Disable or delete default database accounts Apply regular patches / updates to the DB server Use Web Application Firewalls (WAF) to block malicious SQL patterns Conduct periodic privilege audits

Number	Title	Type	Severity	Status	Score	Description	Mitigations
101	New STRIDE threat	Tampering	TBD	Mitigated		Provide a description for this threat	Provide remediation for this threat or a reason if status is N/A