URL shortener (LLM)

Owner: LLM
Reviewer:
Contributors:
Date Generated: Fri Dec 06 2024

OWASP Threat Dragon

Executive Summary

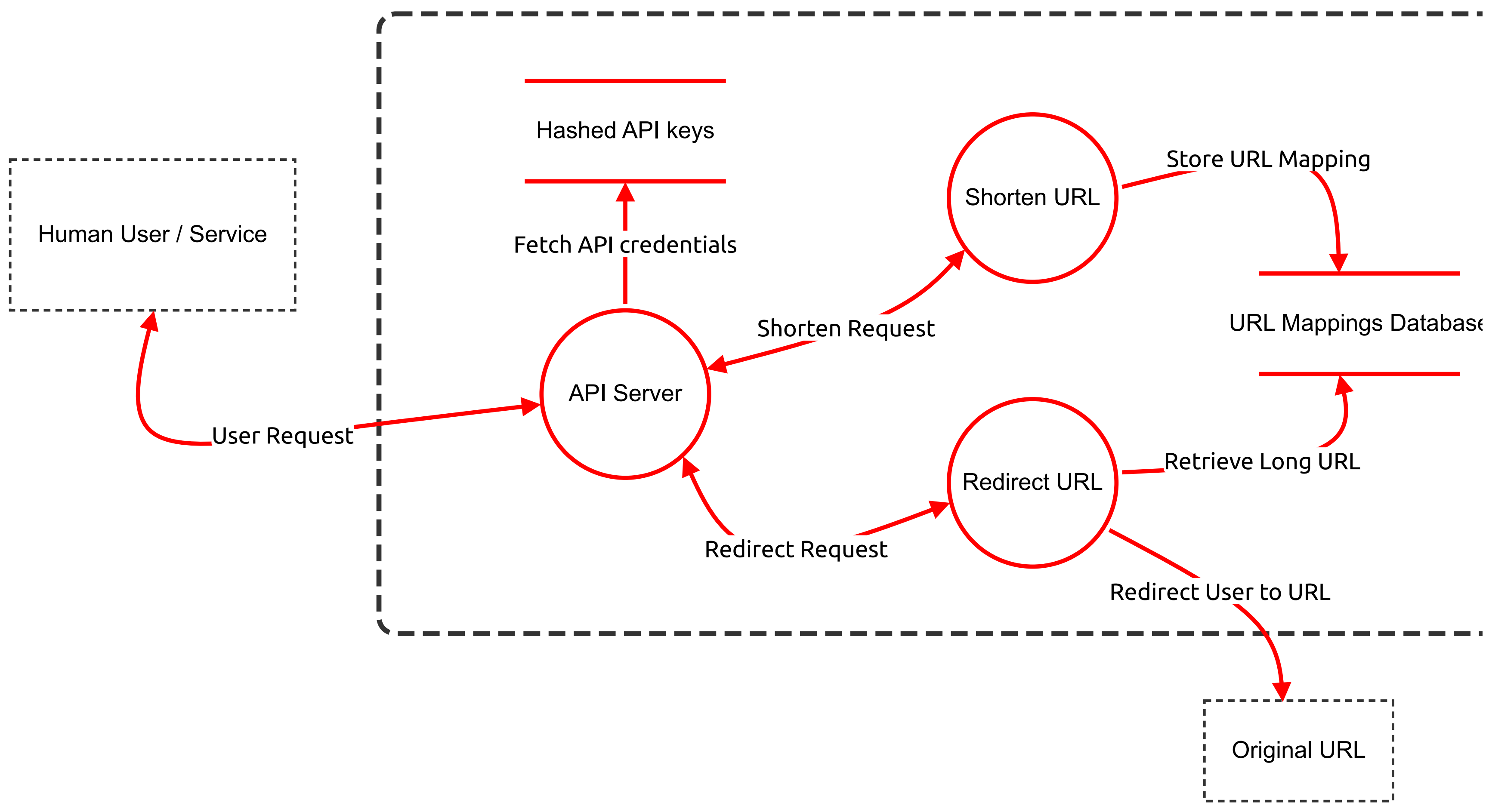
High level system description

The URL Shortener Service aims to provide a simple and efficient way for users to convert long URLs into short, easily shareable URLs. The service ensures functionality, reliability, and security while interacting with users and handling their data.

Summary

Total Threats	75
Total Mitigated	0
Not Mitigated	75
Open / High Priority	40
Open / Medium Priority	34
Open / Low Priority	1
Open / Unknown Priority	0

URL Shortener Service Architecture



URL Shortener Service Architecture

Human User / Service (Actor) *- Out of Scope*

Reason for out of scope: Incoming requests outside of application scope

Description: External services or human users may interact with the API for bulk operations or integrations.

Number	Title	Type	Priority	Status	Score	Description	Mitigations

Redirect URL (Process)

Description: Enables users to resolve short URLs and be redirected to the corresponding long URLs via a dedicated API.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	URL Redirection Manipulation	Tampering	High	Open	9	An attacker might manipulate the redirection process by tampering with short URL requests, potentially redirecting users to malicious sites instead of the intended destination.	Implement input validation and verify the integrity of short URLs using cryptographic techniques. Employ URL monitoring and logging to detect unusual redirection patterns.
2	Unauthorized Access to URL Mappings	Information disclosure	High	Open	8	An unauthorized user could gain access to the URL Mappings Database, leading to exposure of sensitive data such as long URLs and user IDs.	Ensure robust authentication and access control mechanisms for database access. Encrypt sensitive data at rest and in transit.
3	Denial of Service through High Traffic	Denial of service	Medium	Open	7	An attacker could overwhelm the URL redirection service with a large number of requests, leading to service unavailability for legitimate users.	Implement rate limiting and throttling mechanisms on the API server to handle excessive traffic gracefully. Monitor and block malicious IPs using firewalls or network security solutions.
4	Spoofing of API Requests	Spoofing	Medium	Open	6	An attacker might spoof API requests to impersonate a legitimate user, potentially disrupting user experiences or accessing unauthorized resources.	Use strong authentication mechanisms like OAuth2 for API access. Implement TLS for securing data in transit and ensure that API keys are securely managed.
5	Lack of Accountability for API Requests	Repudiation	Medium	Open	6	Users or attackers could deny any action by exploiting insufficient logging of API request activities, complicating security analysis and incident response.	Implement comprehensive logging and monitoring for all API activities, including timestamps, source IP addresses, and user IDs. Consider retaining logs for a reasonable period for audit and forensic purposes.
70	Cross-Site Scripting via URL Redirect	Tampering	High	Open	9	An attacker could craft a malicious URL that when accessed through the redirect process, executes unauthorized scripts in the user's browser context. This could lead to account compromise or unauthorized data access.	Implement strict input validation and sanitization of all URL parameters. Utilize Content Security Policy (CSP) to prevent the execution of unauthorized scripts.
71	Insecure Direct Object Reference	Information disclosure	High	Open	8	An attacker could manipulate short URL IDs via inference or brute force, allowing access to redirect URLs without authorization, leading to potential leakage of sensitive URL mappings.	Use non-predictable, hard-to-guess identifiers for short URLs and ensure all access is subject to authorization checks. Implement rate limits to deter brute force attacks.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
72	Man-in-the-Middle Attack on Redirect Requests	Information disclosure	High	Open	9	An attacker could intercept traffic between the client and API server, modifying redirect requests to point users to a malicious destination.	Enforce HTTPS across all communications to ensure data in transit is encrypted and protected from interception.

Shorten URL (Process)

Description: Allows users to submit long URLs and receive shortened URLs via a dedicated API.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	URL Mapping Tampering	Tampering	High	Open	8	An attacker could manipulate the URL mappings stored in the database, redirecting users to malicious sites instead of their intended destination.	Implement strong access controls and authentication methods for database access. Use database logging and monitoring for unauthorized changes.
2	Sensitive Data Exposure	Information disclosure	Medium	Open	6	Long URLs may contain sensitive parameters or information that could be disclosed if not properly protected.	Ensure that URLs are encrypted in transit using TLS and consider redacting sensitive information before storing.
3	Denial of Service via Spam URL Submissions	Denial of service	Medium	Open	5	An attacker could submit a large number of URLs to the service, overloading the system and preventing legitimate use.	Limit the rate of URL submissions per user and implement CAPTCHA to prevent automated submissions.
4	Unauthorized URL Deletion	Elevation of privilege	Medium	Open	6	An attacker could exploit the system to delete URL mappings that they are not authorized to access.	Implement role-based access controls and require proper authentication and authorization checks before URL deletion.
5	Malicious URL Injection	Spoofing	High	Open	7	An attacker could craft specific URLs with malicious payloads to execute upon redirection, affecting clients.	Validate and sanitize URLs before processing and ensure redirects are to trusted domains.
67	API Endpoint Bruteforce	Denial of service	High	Open	8	Attackers may attempt to exhaustively test possible short URL combinations via the API, possibly uncovering or inferring URLs that contain sensitive information.	Implement rate limiting and logging for repeated access patterns. Use CAPTCHA to ensure interactions are genuine and from a human user.
68	Insufficient Authentication on API Server	Elevation of privilege	High	Open	8	Without proper authentication, an attacker could potentially gain unauthorized access to the API server functions, exploiting the API to perform unauthorized operations like URL shortening or redaction.	Implement strong authentication mechanisms for all API interactions. Use token-based authentication with expirations and apply two-step verification for critical operations.
69	Data In Transit Interception	Information disclosure	High	Open	9	Sensitive data such as long URLs and user identifiers may be intercepted during transit between API server and URL database if not encrypted, potentially leading to information leakage.	Ensure that all data in transit is encrypted using TLS and secure communication channels are enforced.

URL Mappings Database (Store)

Description: Data store containing the mappings between short URLs and their corresponding long URLs.

DATA STORED: Long URL, Short URL, UserID, CreationDate

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Unauthorized Access to URL Mappings Database	Elevation of privilege	High	Open	8	Attackers may gain unauthorized access to the URL Mappings Database to retrieve or modify URL mappings.	Implement strict access controls using authentication and authorization mechanisms. Ensure that only authorized users or services have access to the database.
2	Data Tampering in URL Mappings	Tampering	High	Open	8	An attacker could tamper with stored URL mappings, altering short URLs to redirect to malicious sites.	Apply database integrity controls and hashing mechanisms to verify the integrity of the data. Regularly audit database changes.
3	Lack of Encryption for Sensitive Data	Information disclosure	Medium	Open	6	The absence of encryption for data stored in the URL Mappings Database may lead to unauthorized data access or interception.	Encrypt sensitive information, such as URL mappings, at rest and in transit using strong encryption standards.
4	Denial of Service through Database Overload	Denial of service	Medium	Open	5	An attacker could flood the URL shortener service with requests, overwhelming the database and leading to denial of service.	Implement rate limiting and request throttling at the API level to prevent excessive requests. Monitor traffic patterns for anomalies.
5	SQL Injection Attacks on URL Mappings Database	Tampering	High	Open	8	An attacker may exploit weaknesses in database query handling to perform SQL injection, compromising the data's integrity and confidentiality.	Use prepared statements and parameterized queries to interact with the database. Conduct regular security audits and input validation.
73	API Misconfiguration Leading to Unauthorized Data Exposure	Information disclosure	High	Open	8	Improper configuration or lack of security controls in the API endpoints might allow attackers to bypass authentication and access or modify URL mappings.	Ensure API Gateway or server-side protections enforce strict authentication and authorization controls. Regularly scan and test APIs for security misconfigurations.
74	Brute Force Attack on Short URLs	Denial of service	Medium	Open	6	Attackers could perform brute force attacks to guess short URLs and gain unauthorized access to the original URLs or related information.	Implement rate limiting on URL access attempts and use longer or more complex URL tokens to mitigate brute force attacks.
75	Cross-Site Scripting (XSS) through Redirection URLs	Tampering	High	Open	9	An attacker could exploit poorly validated input in URL redirection functionality to inject malicious scripts.	Validate and sanitize all inputs, ensuring that no scripts can be injected through URL parameters. Use Content Security Policy (CSP) headers to mitigate XSS risks.

Original URL (Actor) *- Out of Scope*

Reason for out of scope: Original URL not in scope of URL shortener application

Description: Destination server of the original URL

Number	Title	Type	Priority	Status	Score	Description	Mitigations

API Server (Process)

Description: The API Server is the central component of the URL Shortener Service. It manages all interactions with users and external services, handling requests for both shortening URLs and redirecting from short URLs to their original long URLs.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Unauthorized API Access	Spoofing	High	Open	9	An attacker could exploit vulnerabilities to bypass API authentication mechanisms by intercepting or tampering with the hashed API key exchange.	Implement secure transmission protocols like TLS to protect data in transit. Utilize multi-factor authentication (MFA) and regularly rotate API keys.
2	Malicious URL Injection	Tampering	High	Open	8	An attacker can inject a malicious long URL during the shortening process, potentially leading users to phishing sites or distributing malware.	Implement strict URL validation and filtering before shortening any URL. Use a real-time threat intelligence service to block known malicious URLs.
3	Data Breach of User Information	Information disclosure	High	Open	9	Sensitive user information, such as API keys and URLs, could be exposed if the database storing this information is compromised.	Encrypt sensitive data at rest and implement strict access controls. Use database activity monitoring to detect suspicious access.
4	API Usage Spoofing	Repudiation	Medium	Open	7	Attackers could send a large number of requests to the API, using valid but leaked API keys to mask their identity, affecting legitimate user experience.	Monitor API usage patterns for anomalies. Implement rate limiting and alert users of unusual activity related to their API keys.
5	Denial of Service via URL Redirection	Denial of service	Medium	Open	6	An attacker could generate a flood of redirection requests leading to an overload of the API server, affecting its availability.	Implement rate limiting and request throttling. Use Content Delivery Networks (CDNs) to absorb and mitigate high request volumes.
61	Cross-Site Scripting (XSS) in Short URL Handling	Tampering	High	Open	9	An attacker could exploit vulnerabilities in the user input fields to inject malicious scripts, potentially executing them in the context of the API Server, leading to information disclosure or user impersonation.	Implement input sanitization and validation on all user inputs. Use Content Security Policy (CSP) headers and escape user data when rendering.
62	Excessive Data Exposure	Information disclosure	Medium	Open	7	The API might expose more data than necessary in its responses. For example, returning unnecessary user data when fulfilling requests could lead to information disclosure.	Minimize data returned by the API to only what is strictly necessary. Implement principles of least privilege and security through obscurity.
63	Lack of Rate Limiting on Shorten URL Process	Denial of service	Medium	Open	6	Without proper rate limiting, an attacker could flood the API with URL shortening requests, potentially leading to service degradation and abuse of resources.	Implement rate limiting and request throttling based on IP and API keys to prevent abuse. Set appropriate thresholds for the number of requests allowed.

Hashed API keys (Store)

Description: Database containing hashed API keys for authentication purposes.

DATA STORED: Hashed API Key, UserID

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Weak Hashing Algorithm for API Keys	Information disclosure	High	Open	9	The database might use a weak hashing algorithm for storing API keys, making it vulnerable to brute-force attacks.	Use a strong, industry-standard hashing algorithm like bcrypt to store API keys and regularly audit the strength of hashes.
2	Unauthenticated Access to Hashed Keys	Information disclosure	High	Open	8	Potential unauthorized access to the database containing hashed API keys could lead to information disclosure.	Implement strong access controls and authentication mechanisms to ensure only authorized entities can access the database.
3	Lack of Encryption in Data Storage	Information disclosure	Medium	Open	7	The data in the database, including hashed API keys, is not encrypted, which could lead to potential data compromise if the database is accessed.	Encrypt data at rest to ensure that even if unauthorized access occurs, the data remains protected.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
4	Tampering with Stored API Keys	Tampering	Medium	Open	6	An attacker could potentially tamper with the stored hashed API keys to disrupt the service or gain unauthorized access.	Implement integrity checks using digital signatures or HMAC to ensure that API key data has not been tampered with.
5	Denial of Service via Exploitation of Authentication Mechanism	Denial of service	Medium	Open	5	A malicious actor might exploit the authentication system tied to the hashed API keys through excessive requests, leading to performance bottlenecks or a denial of service.	Implement rate limiting and use CAPTCHAs to mitigate automated requests that could lead to overloading the system.
64	Spoofing API Server Requests	Spoofing	High	Open	9	An attacker might impersonate the API Server to send unauthorized requests to the database containing hashed API keys, potentially facilitating unauthorized access or denial of service attacks.	Implement mutual TLS (mTLS) to ensure that only authenticated and verified servers can communicate with the database, and regularly audit access logs.
65	Replay Attacks on API Credential Fetch	Information disclosure	Medium	Open	7	An attacker could capture the traffic between the API Server and the database when API credentials are fetched and replay it to gain unauthorized access.	Use unique nonces and timestamps with each request to ensure requests are fresh and cannot be replayed.
66	Network Eavesdropping on Data Flows	Information disclosure	Medium	Open	7	Eavesdroppers on the network could intercept the data flow between the API Server and the database, potentially capturing hashed API keys during transmission.	Implement end-to-end encryption such as HTTPS/TLS to secure data in transit and prevent unauthorized interception.

Store URL Mapping (Data Flow)

Description: The API Server processes the request for shortening a URL, generates a short URL, and stores the mapping between the short URL and the long URL in the URL Database.

DATA EXCHANGED: Short URL, Long URL

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Unauthorized Access to URL Mapping	Elevation of privilege	High	Open	9	An attacker could gain unauthorized access to the URL mappings stored in the database, potentially allowing them to alter or delete mappings.	Implement proper access controls and authentication mechanisms for the database to ensure only authorized personnel can access or modify URL mappings.
2	Data Interception in Transit	Information disclosure	High	Open	8	Without encryption, the data exchanged between the API Server and the URL Database could be intercepted by an attacker, leading to sensitive information disclosure.	Use TLS to encrypt data in transit between the server and the database to prevent interception.
3	URL Mapping Tampering	Tampering	High	Open	8	An attacker might tamper with the stored URL mappings, redirecting short URLs to malicious or incorrect destinations.	Implement checksum or hash verification to ensure URL mappings have not been tampered with.
4	Denial of Service via URL Flooding	Denial of service	Medium	Open	6	An attacker could flood the service with requests to shorten URLs, overwhelming the system resources and denying service to legitimate users.	Implement rate limiting and CAPTCHAs to mitigate automated and excessive requests.
5	Insecure URL Input Handling	Tampering	Medium	Open	6	The system may lack validation on input URLs, potentially allowing script or command injection attacks.	Implement strict input validation and sanitation to ensure only valid URLs are processed.

Shorten Request (Data Flow)

Description: The user submits a long URL to the API for shortening.

DATA EXCHANGED: Short URL, Long URL, UserID

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Man-in-the-Middle Attack on Shorten Request	Tampering	High	Open	9	An attacker could intercept the communication between the user and the API server since the data flow is not encrypted, potentially altering or stealing data such as URLs and UserIDs.	Implement end-to-end encryption for all data exchanged, such as using HTTPS for secure communication.
2	Unauthorized URL Submission	Elevation of privilege	Medium	Open	7	An attacker submits malicious URLs for shortening, which could be used in phishing attacks or to distribute malware.	Implement authentication mechanisms to ensure that only authorized users can submit URLs for shortening.
3	Information Disclosure of User Data	Information disclosure	High	Open	8	Sensitive user data, such as UserIDs, could be accessed by unauthorized parties if proper access controls are not enforced.	Enforce strict access controls and use least privilege principles to restrict data access based on user authentication and authorization.
4	Denial of Service via Excessive Shorten Requests	Denial of service	Medium	Open	6	An attacker could overwhelm the API server with a flood of URL shortening requests, potentially leading to service disruption.	Implement rate limiting and IP blacklisting to prevent excessive requests from overloading the server.
5	Lack of Input Validation on URL Submission	Tampering	Medium	Open	7	If there is insufficient input validation, malicious users can exploit the service by submitting invalid or harmful URLs for shortening.	Implement rigorous input validation to ensure all submitted URLs comply with expected formats and criteria.

Redirect Request (Data Flow)

Description: The user submits a short URL to the API and gets redirected to the original Long URL.

DATA EXCHANGED: Short URL, Long URL

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Spoofed Redirect URLs	Spoofing	High	Open	8	An attacker could generate a malicious short URL that mimics a legitimate service to deceive users into visiting a harmful website.	Implement URL validation checks and warning messages for users when redirecting to non-whitelisted domains.
2	Tampering with Redirect Requests	Tampering	High	Open	9	An attacker may intercept and alter the short URL redirect requests to direct users to a different, potentially malicious, long URL.	Encrypt all data in transit using SSL/TLS to prevent interception and tampering.
3	Information Disclosure through Unencrypted Flow	Information disclosure	Medium	Open	7	Sensitive data (short and long URLs) could be exposed during transmission as the redirect flow is not encrypted.	Implement SSL/TLS encryption for all communication between the client and the server.
4	Denial of Service via Resource Exhaustion	Denial of service	High	Open	8	An attacker could flood the API with redirect requests, causing excessive load and making the service unavailable to legitimate users.	Implement rate limiting and request throttling to manage incoming requests effectively and prevent abuse.
5	Insufficient Authentication on Redirect Requests	Elevation of privilege	Medium	Open	6	Lack of proper authentication checks could allow unauthorized users to misuse the redirect service.	Implement authentication checks to verify user identity before processing redirect requests.

Redirect User to URL (Data Flow)

Description: The user is redirected to the Long URL.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	URL Tampering by Malicious Actor	Tampering	High	Open	9	A malicious actor might intercept the request and tamper with the URL, redirecting the user to a harmful site.	Implement strong validation and integrity checks on the URL redirection process. Use HTTPS to encrypt traffic.
2	Malicious URL Shortening	Information disclosure	High	Open	8	Users could intentionally shorten URLs that lead to malicious sites and distribute them, posing a risk to unsuspecting users.	Incorporate URL scanning and filtering against known harmful sites. Implement user reputation scoring.
3	Denial of Service through Excessive Requests	Denial of service	Medium	Open	7	An attacker could flood the redirect service with excessive requests, leading to service degradation or downtime.	Use rate limiting and CAPTCHA mechanisms for users requesting redirects.
4	Insecure Redirection Protocol	Information disclosure	Medium	Open	6	The service may use an unencrypted protocol for redirection, making the data susceptible to interception.	Ensure that the redirection uses HTTPS to encrypt all data in transit.
5	Lack of Logging for User Redirections	Repudiation	Low	Open	5	Absence of logging could hinder tracing activities and identifying misuse or breaches.	Implement comprehensive logging of redirect requests with analysis for anomalies.

Fetch API credentials (Data Flow)

Description: Hashed API keys are pulled to authenticate the user before performing any operation.

DATA EXCHANGED: Hashed API key

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Unauthorized Access to API Keys	Elevation of privilege	High	Open	9	An attacker might gain unauthorized access to the storage where hashed API keys are stored if proper access controls are not implemented.	Implement strong access controls and monitoring on the database where API keys are stored. Ensure principle of least privilege is enforced.
2	Weak Hashing Algorithm for API Keys	Information disclosure	High	Open	9	If a weak hashing algorithm is used for API keys, it could be vulnerable to attacks that recover the original API key from the hash.	Use a strong, cryptographic hash function like SHA-256 for hashing API keys. Regularly audit and update cryptographic practices.
3	API Key Replay Attack	Tampering	Medium	Open	7	An attacker could intercept and reuse a valid hashed API key if the transmission is not secure.	Implement encryption (e.g., TLS) for data in transit to protect API keys from being intercepted and reused.
4	Logging of API Keys in Plaintext	Information disclosure	Medium	Open	6	If hashed API keys are logged in plaintext during the transmission process, they could be exposed to unauthorized access.	Ensure that logging practices do not include sensitive information like API keys. Implement log scrubbing and access controls.
5	Man-in-the-Middle Attack on API Credential Fetch	Spoofing	High	Open	8	An attacker could intercept communications between the client and the server to alter or steal API credentials.	Use secure communication protocols, like TLS, to prevent interception and manipulation of data during transmission.

User Request (Data Flow)

Description: The User sends a request to the API Server to shorten a long URL or to get the redirect URL.

DATA EXCHANGED: Long URL, Short URL, API key

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Man-in-the-Middle Attack on User Request	Tampering	High	Open	9	An attacker could intercept user requests to the API Server to either shorten a URL or fetch a redirect URL due to lack of encryption, potentially modifying or accessing sensitive data like the API key.	Implement HTTPS to encrypt data in transit, ensuring the confidentiality and integrity of the data exchanged between the user and the API server.
2	API Key Leakage due to Insecure Transmission	Information disclosure	High	Open	9	As the data exchanged includes an API key and isn't encrypted, there is a risk of this key being intercepted by unauthorized entities, leading to unauthorized access or abuse of the URL shortener API.	Encrypt all sensitive information using TLS/SSL during transmission. Implement API key rotation and usage monitoring to detect and limit exposure.
3	Denial of Service through API Overuse	Denial of service	Medium	Open	7	An attacker could overload the API Server with requests by exploiting the user request flow, potentially leading to service disruption and denial of service for legitimate users.	Implement rate limiting and request throttling to prevent excessive requests from a single source, and monitor unusual activity patterns to quickly identify and defend against attacks.
4	Spoofing via API Key Misuse	Spoofing	Medium	Open	6	An adversary could use a valid API key that was obtained through insecure transmission or sharing to impersonate a legitimate user, gaining unauthorized access or performing actions on behalf of the user.	Implement strong API authentication mechanisms, such as OAuth, and track API key usage to ensure that keys are only used by authorized users or services.
5	Replay Attacks in User Requests	Repudiation	Medium	Open	6	An attacker might capture and replay user requests to the API Server, effectively duplicating operations like URL shortening without the user's consent.	Include nonces or timestamps in requests to ensure that each request is unique and validate these elements in the API server to detect and prevent replay attacks.

Retrieve Long URL (Data Flow)

Description: Long URL mapping is pulled from the database in order to redirect user.

DATA EXCHANGED: Short URL, Long URL, UserID

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	Unencrypted Data Transmission	Information disclosure	High	Open	8	The retrieval of the long URL from the database is not encrypted, leading to potential interception of sensitive information.	Ensure encryption is used for data transmission, such as using TLS/SSL.
2	Unauthorized Access to URL Mappings	Elevation of privilege	High	Open	9	An unauthorized user may access the URL mappings database, potentially exposing sensitive data such as user IDs and URLs.	Implement strong access controls and authentication mechanisms to prevent unauthorized access.
3	Short URL Enumeration	Information disclosure	Medium	Open	6	Attackers could systematically send requests to guess valid short URLs and access the associated long URLs.	Implement rate limiting and use of more complex short URL generation to mitigate enumeration.
4	Database Tampering	Tampering	High	Open	9	An attacker could tamper with the database storing URL mappings, altering the entries to redirect traffic elsewhere.	Implement measures like data integrity checks and logging of database transactions.
5	Denial of Service through Redirect Overload	Denial of service	Medium	Open	7	Massive requests for redirection could overwhelm the service leading to legitimate users being unable to access it.	Implement rate limiting and distributed denial of service (DDoS) protection mechanisms.