URL shortener (Engineer 1)

Owner: Engineer 1
Reviewer:
Contributors:
Date Generated: Wed Dec 25 2024

OWASP Threat Dragon

Executive Summary

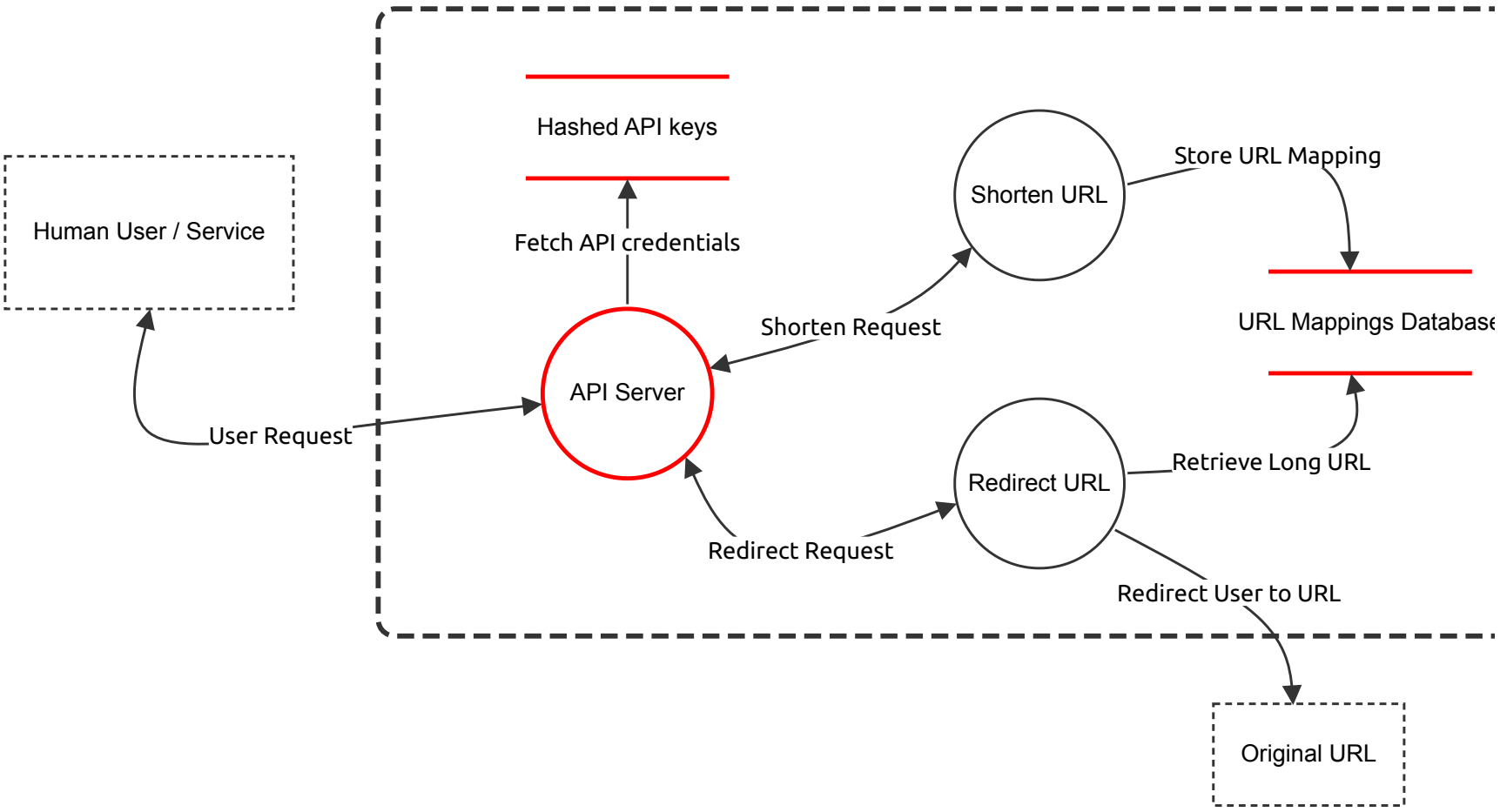
High level system description

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	14
Total Mitigated	0
Not Mitigated	14
Open / High Priority	5
Open / Medium Priority	5
Open / Low Priority	4
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

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

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
15	Injection of malitious links as target URLs	Tampering	Medium	Open		A malitious actor could inject malitious links as target URLs directly to database if the API Server auth is bypassed	Implement additional integrity checks in Database by introducing new field with entry's hash.
16	User's can modify other user's database entries	Tampering	Medium	Open		Other user's keys can be used to manage other user's records without RBAC in place.	Implement Role-Based Access Control (RBAC) and Least Privilege by limitting API key permissions to only the necessary actions for the specific client or service.
28	No database access Logging in place	Repudiation	Low	Open		Lack of the informations about how, when and who modified database entry.	Enable Database Access Logging

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
20	Server overload from a high volume of requests	Denial of service	High	Open		Server is prone to DoS and DDoS attacks if malitious actor decides to send large number of requests to API Server	Implement rate limiting and thorttling on the API to limit the number of requests per minute. Use firewalls and DDoS protection mechanisms like IP Whitelisting and Geo-Restrictions.
21	Injection and Data Corruption	Tampering	High	Open		Malitious actor can send unexpected or harmful data in API requests. When an API accepts data without validating its structure and format, malformed or incorrect data can end up in the database, leading to data integrity issues.	Implement request body validation sanitization and format checks to accept only valid data.
22	Server Errors due to Invalid Data	Denial of service	Medium	Open		Unvalidated or improperly formatted data can cause exceptions or errors when processed by the server. Attackers could send requests with extremely large or deeply nested payloads, consuming excessive server resources during parsing or processing.	Implementing request body validations, sanitizations, limits.
23	Bruteforce	Spoofing	Medium	Open		Attackers could attempt to guess the API Key by sending multiple requests to an API Server.	Ensure that API keys are sufficiently long and difficult to guess. Implement Rate Limiting and Throttling
24	API Key stored insecurily on client side	Spoofing	High	Open		If the API Key is not stored securily on the client side it can be prone to leackage and utilization by malitious actors to gain access to the system.	Issue short-lived access tokens instead of long-lasting API keys. Use OAuth2.0 and JWTs that expire after a short period. This reduces the window of time during which a compromised token can be used. Attackers will need to continually re-obtain new tokens, which can be difficult if proper authentication mechanisms are in place. Implement Role-Based Access Control (RBAC) and Least Privilege. Ensure you can quickly revoke or rotate API keys/tokens if they are suspected to be compromised.
25	API Key Leakage in transit	Spoofing	High	Open		A MITM attack can be executed to retrieve the value of API Key if the data encryption in transit is not implemented	Ensure data in transit encryption by implementing TLS.
26	Lack of request logging	Repudiation	Low	Open		Users may deny performing certain operations (e.g., creating a redirect)	Log all operations related to users, such as creating and deleting URLs, and verify operations.
27	Attacker can read all resources	Information disclosure	High	Open		By lack of RBAC attacker can manage all database entries using the single API Key	Implement Role-Based Access Control (RBAC) and Least Privilege

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
29	No database access Logging in place	Repudiation	Low	Open		Lack of the informations about how, when and who modified database entry.	Enable Database Access Logging
31	Database data could leak due to misconfiguration	Tampering	Low	Open		If database instance or its auth is misconfigured hashed API Keys could leak	Encrypt API keys and other sensitive data in the database. Ensure that database access is restricted to authorized components/users only.
32	Hash collisions due to weak function	Information disclosure	Medium	Open		If a weak hashing function (e.g., MD5 or SHA-1) is used, an attacker could potentially generate collisions or alter data without detection. Weak hashing functions are more vulnerable to bruteforce or rainbow table attacks, which allow attackers to more easily reverse or guess the hashed values of API Keys.	Replace weak hash functions like MD5 or SHA-1 with stronger, modern alternatives, such as SHA-256 or SHA-3.

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

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

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

Redirect User to URL (Data Flow)

Description: The user is redirected to the Long URL.

Number	Title	Type	Priority	Status	Score	Description	Mitigations

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

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

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