URL shortener (Engineer 5)

Owner: Engineer 5
Reviewer:
Contributors:
Date Generated: Mon Dec 16 2024

Executive Summary

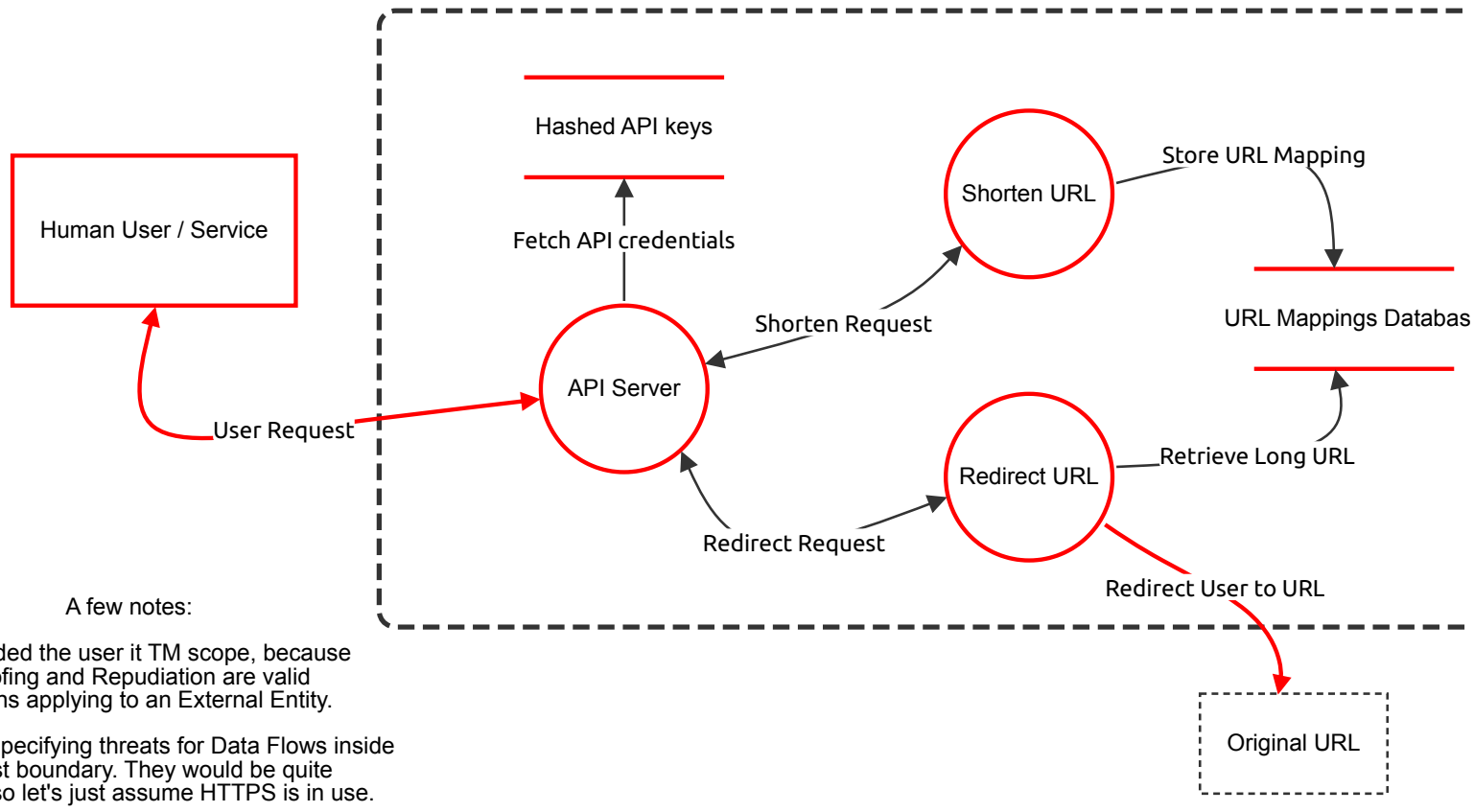
High level system description

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	25
Total Mitigated	0
Not Mitigated	25
Open / High Priority	8
Open / Medium Priority	6
Open / Low Priority	11
Open / Unknown Priority	0

URL Shortener Service Architecture



A few notes:

1. I included the user it TM scope, because Spoofing and Repudiation are valid concerns applying to an External Entity.

2. I skipped specifying threats for Data Flows inside the trust boundary. They would be quite repetitive, so let's just assume HTTPS is in use.

. I would rather depict API Server/Shorten URL/Redirect URL blocks as a single Process block, or maybe include the latter two under "URL Shortener" block.

4. Since not much is said about mitigations that are alredy in place, I conjured some potential ones in form of ıostly) ideas on how certain problems should be approached. There are some cautions about popular misconceptions as well.

5. Risk decisions are hinted at in the remediations mostly just because I believe it would be beneficial for some functionality to be dropped or transferred, but that might probably be a bit of an extrapolation of the exercise rules.

URL Shortener Service Architecture

Human User / Service (Actor)

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

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	General Spoofing: Impersonation	Spoofing	High	Open		An attacker impersonates a user account	Proposed action: Mitigate System description states that users authenticate against the service. Service should implement measures to assure users are correctly authenticated and authorised. It should be specified if different roles are needed and if permissions should be applied on a resource level (e.g. roles for only resolving short URLs, roles allowed resolving only particular URLs, etc.)
2	General Repudiation: Abusive Activity	Repudiation	Low	Open		A user denies abusive activity	Proposed action: Accept the risk This action is probably not very valuable, but still some minimal non repudiation assurances could be implemented. Review if the service provides enough evidence (e.g. access logs) which could help identify offensive users.

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
19	Spoofing: Malicious Redirection By Impersonation	Spoofing	Low	Open		An attacker spoofs redirection API in order to point users to bogus URLs	This should be already covered by mitigations provided for the "API Server" block.

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
16	Tampering: Overwriting Existing Data	Tampering	High	Open		An authenticated attacker or user tries overwriting an existing URL mapping	Proposed action: Mitigate Overwriting an existing mapping should not be allowed. If it's a requirement, then only limited roles should be able to do that, and the changes should be only done in incremental manner, so that the database retains historical destinations (which contributes to non-repudiation evidence). It might be helpful to notify users that they're being redirected to a new address and only stop a given notification if they wish for it to stop (e.g. "Don't show this warning again" checkbox).

Number	Title	Type	Priority	Status	Score	Description	Mitigations
33	Tampering: Potential Nondeterminism	Tampering	Low	Open		An attacker provides conflicting data in bulk requests	Proposed action: Mitigate Two cases: 1. Some roles are allowed to specify short ids. If bulk request contains clashing definitions, the request should be rejected. 2. Usual shorten operations should only accept long URL parameter and return randomised short URLs. This way clashes are not possible.
36	Spoofing: Typosquatting	Spoofing	Medium	Open		An attacker crafts requests to produce short URLs similar to already existing ones, with an intention to lure legitimate users to bogus destinations.	Proposed action: Mitigate Users (or not all of them) should not be able to specify resulting short URLs. The system should generate cryptographically random identifiers, which must not be consecutive or easy to predict during generation time, so that risk of typos is diminished.

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
7	Tampering: Malicious Redirection By Permanent Change	Tampering	High	Open		An attacker modifies destination URL(s)	Proposed action: Mitigate Database access should be protected to specific roles. Additionally data should be structured in a way, so that (if possible) no role can modify and existing mapping while losing the old value permanently. Every entry might be cryptographically "chained", so that each row contains a hash of itself and hash of the previous row. This way integrity of whole database is assured to reasonable extent. The first row might contain a secret value, so that full history rewrite is not possible without knowing it.
8	General DoS: Data Deletion	Denial of service	Medium	Open		An attacker removes existing mappings	Proposed action: Mitigate Depending on how the data is structured it might be possible to disallow any role from removing existing rows. Vandalism can be mitigated by backups, to a certain extent.
10	Information Disclosure: User Activity Analysis	Information disclosure	Low	Open		An attacker learns identities of users who created particular mappings	Proposed action: Mitigate or Accept the risk Aside from securing DB access, it's probably unnecessary to keep submitting user ids in plaintext. Maybe computing hashes/keys to track owners/submitters internally would be sufficient? If URL themselves are not very sensitive or revealing interesting info in any way, then possibly the risk is minimal and can be accepted without much mitigation (that is, apart from regular access control on the DB).
24	General DoS	Denial of service	Low	Open		An attacker floods database with excessive number of operations, or operations that are computationally expensive	Proposed action: Mitigate Database should not be publicly accessible. Consider implementing rate limiting mechanisms, limiting allowed operations, etc.
28	General Repudiation: DB Operations	Repudiation	Low	Open		An attacker denies an action performed directly on the database	Proposed action: Mitigate "Unusual" DB operations, including admin access, should be logged. Logs/evidence should be stored long enough to be able to reason about a potential incident

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
11	Spoofing: Credentials Capture and Activity Analysis	Spoofing	High	Open		An attacker spins up a malicious URL Shortener service in order to capture user credentials and/or analyse users' activity	Proposed action: Mitigate Service should be authenticated with a TLS certificate, so that users are sure to connect to the right thing. Minimal recommended TLS version is 1.2
12	General Expansion of Privilege: Injection	Elevation of privilege	High	Open		An attacker crafts a malicious input in order to perform a database injection attack	Proposed action: Mitigate Review and implement adequate user input validation mechanisms. Assume that user input is not trusted.
13	General DoS	Denial of service	High	Open		An attacker floods service with excessive number of requests, or requests that are computationally expensive	Proposed action: Mitigate Introduce suitable mechanisms like rate limiting, WAF, etc. Circuit breaker might be useful if there's a concern of affecting subsequent systems (like databases). Consider allowing only subset of DB operations and enforce parameter validation, so that it's harder for the attacker to craft computationally expensive parameters.
26	General Expansion of Privilege: System Process Takeover	Elevation of privilege	High	Open		An attacker takes over the system process in order to abuse its overall operation	Proposed action: Mitigate or Avoid Mitigate: Assure OS/platform is secured and up-to-date. Limit administrative access to internal network only, or Avoid: Consider using cloud managed (serverless?) services in order to make the attack area smaller.
27	General Repudiation: Admin	Repudiation	Medium	Open		An attacker or a malicious insider denies performing an administrative operation	Proposed action: Mitigate Keep track of sign ins and operations performed on the system, if applicable.

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
6	Information Disclosure: Offline Cryptanalysis	Information disclosure	High	Open		Attacker captures hashes from the database in order to analyse/break them offline.	Proposed action: Mitigate or Transfer Database should not be publicly accessible. Access should require mutual authentication, preferably with usage of private keys/certificates. Only specific roles should be allowed access. In order to make offline cryptanalysis hard, "hashes" should be results of key derivation function suitable for usage with passwords, like Argon2 or scrypt. Avoid using hashing algorithms (e.g. SHA family). OR It might be better to not implement this in the service itself and rather use external sign in provider (e.g. OAuth2.0 based) provided by your cloud provider or your organisation.
18	Tampering: Authorisation Mechanism Abuse	Tampering	Medium	Open		An attacker modifies or inserts their own entry in order to create working authorised credentials.	Proposed action: Mitigate or Transfer Protect DB access by only allowing specific roles to add entries. "Hashes" may not be dependent only on user input. Consider including a 'pepper' value (this might be supported by the algorithm like Argon2 natively). OR Switch to external sign in provider.
20	General Repudiation: DB Operations	Repudiation	Low	Open		An attacker denies an action performed directly on the database	Proposed action: Mitigate "Unusual" DB operations, including admin access, should be logged. Logs/evidence should be stored long enough to be able to reason about a potential incident
22	General DoS	Denial of service	Low	Open		An attacker floods database with excessive number of operations, or operations that are computationally expensive	Proposed action: Mitigate Database should not be publicly accessible. Accept only specific operations from the role used by the service.
25	DoS: Authorisation Data Deletion	Denial of service	Medium	Open		An attacker removes legitimate data in order to lock users out	Proposed action: Mitigate or Transfer Deletion should only be allowed for admin roles. Backups might not be recommended, as they would increase Information Disclosure risk. It would be preferable if users can reprovision their access, if the process allows for doing it easily and reliably (i.e. another system could be used as a root of trust) OR Make it an external sign in provider problem :)

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
31	Information Disclosure: User Activity Analysis	Information disclosure	Medium	Open		An attacker eavesdrops on which URLs are visited by users	Server response probably isn't asynchronous, so this should already be covered by enabling TLS, as explained on "User Request" data flow.
32	Tampering: Malicious Redirection By Modifying Response	Tampering	Low	Open		An attacker modifies responses emitted by the service in order to redirect users to bogus URLs	Server response probably isn't asynchronous, so this should already be covered by enabling TLS, as explained on "User Request" data flow.

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
3	General Tampering: MITM	Tampering	Low	Open		An attacker modifies data sent out by the client in order to create a link directed to a different URL.	Proposed action: Mitigate Clients should connect via HTTPS. Minimal recommended TLS version is 1.2.

Number	Title	Type	Priority	Status	Score	Description	Mitigations
5	General Information Disclosure: Credentials Capture via MITM	Information disclosure	Low	Open		An attacker captures credentials from user's request Note: The payload itself (URLs) probably isn't too sensitive, but any risk action concerning credentials will likely apply to payload as well.	Proposed action: Mitigate Clients should connect via HTTPS. Minimal recommended TLS version is 1.2.

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