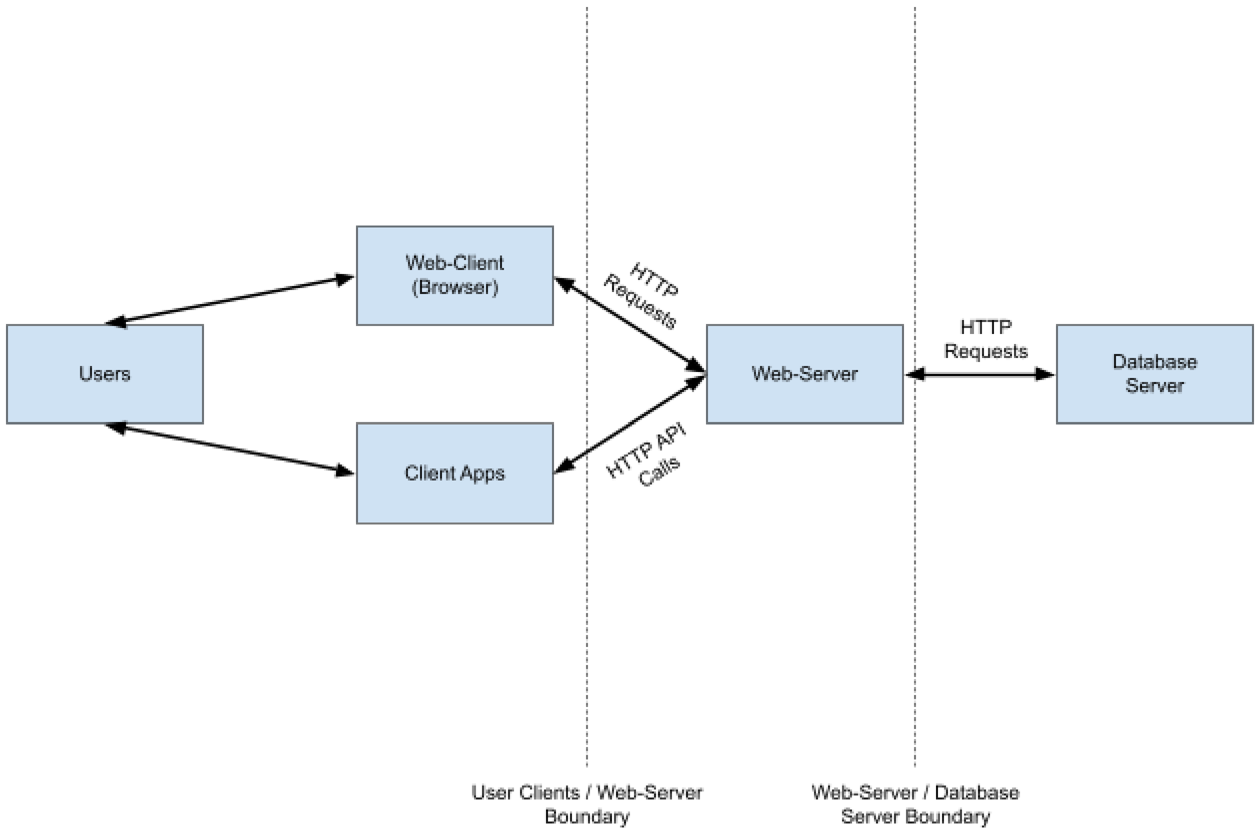
Nathan Hayes-Rich, Peter McCrea, stride.pdf

- **Spoofing,** or impersonating another person or computer, which violates *authenticity*
    - Threat: Only certain people should be able to log into the service, and someone not allowed might end up logging in. Mitigation: Enforce a logical and secure password policy when users initially create their accounts (such that passwords can not be brute forced or easily guessed by dictionary attacks). Additionally, utilize 2FA to prevent spoofers from accessing the services even with stolen login credentials.
    - Threat: Only certain (verified) client apps should be able to interact with the HTTP-based API on the web server. Someone may impersonate a client app. Mitigation: Issue keypairs for verified client apps that get checked on any API call. Keypairs are changed regularly, so if a keypair is compromised it will not be useful for long.
- **Tampering** with data, which violates *integrity*
    - Threat: A lemur wants to overwrite the locations of reported lemur sightings on the database server by hacking into it directly to obfuscate their location. Mitigation: The web servers can only write values to the database server (not delete or alter data), and to log into the database server itself you will need a correct key (e.g. SSH keys which are kept physically secure by administrators).
    - Threat: A lemur intercepts a database write call and changes the call to write the data it wants into the database server. Mitigation: Interactions between the webserver and the database server should use an encrypted session for data transfer (e.g. HTTPS) and the web server should utilize a certificate so end users know they are really talking to the server and not a person/lemur in the middle.
- **Repudiation,** or making it impossible to link an action you performed to you, which violates *non-repudiability*
    - Threat: A lemur may write many malicious fake locations to the database anonymously. Mitigation: With every API call or webclient-webserver interaction to write something to the database, hash the user's username or access key and store that information alongside the actual information, so at a later date you can rehash any suspicious usernames to see the data written by them. Additionally, we force encryption of the message's digest using the user's secret key, thus ensuring any sensical API request will be linked to the user (as long as they have kept their secret key secret).
    - Threat: A lemur may pay a system administrator to delete locations of lemurs from the database server. Mitigation: For each action performed by a system administrator to the web server or the database, the system keeps an append-only log that associates any changes with a user.
- **Information disclosure**, which violates *confidentiality*
    - Threat: An eavesdropper may listen in on HTTP traffic. Mitigation: Use HTTPS for any web client / HTTP API requests sent.
    - Threat: If the database server and webserver aren't on the same local network (and also use HTTP for webserver - database server interactions), an

eavesdropper can listen in on or change the traffic. Mitigation: Again, use HTTPS (potentially with a secure pre-shared key) for communication to and from the database server.

- **Denial of service,** which violates *availability*
    - Threat: Lemurs send lots of malicious API / web-client requests to the webserver, preventing actual users from interacting. Mitigation: Monitor for unusual sets of API / webclient requests, and potentially ignore high volumes of (duplicate) requests from individual IP addresses.
    - Threat: DDOS from a highly funded lemur organization with access to lots of computers. Mitigation: Blacklisting known lemur IP addresses, and potentially only allowing specifically vetted IP addresses from known good users from interacting with the web server.
- **Elevation of privilege,** which violates *authorization*
    - Threat: Lemurs might brute force weak passwords for the database server / web server. Mitigation: Enforcing a password policy such that strong passwords are required. Additionally, prevent multiple failed access attempts.
    - Threat: Highly organized lemur phishing campaign sneaks a piece of malware onto the database server / webserver. Mitigation: Information campaigns on phishing. Enforce security policies on computers with secure access to the database server / web server to prevent downloading something unapproved (say, from an email) directly onto a computer.
- **Data Flow Diagram**


Web-Client (Browser)
Users
Client Apps
HTTP Requests
HTTP API Calls
Web-Server
HTTP Requests
Database Server
User Clients / Web-Server Boundary
Web-Server / Database Server Boundary