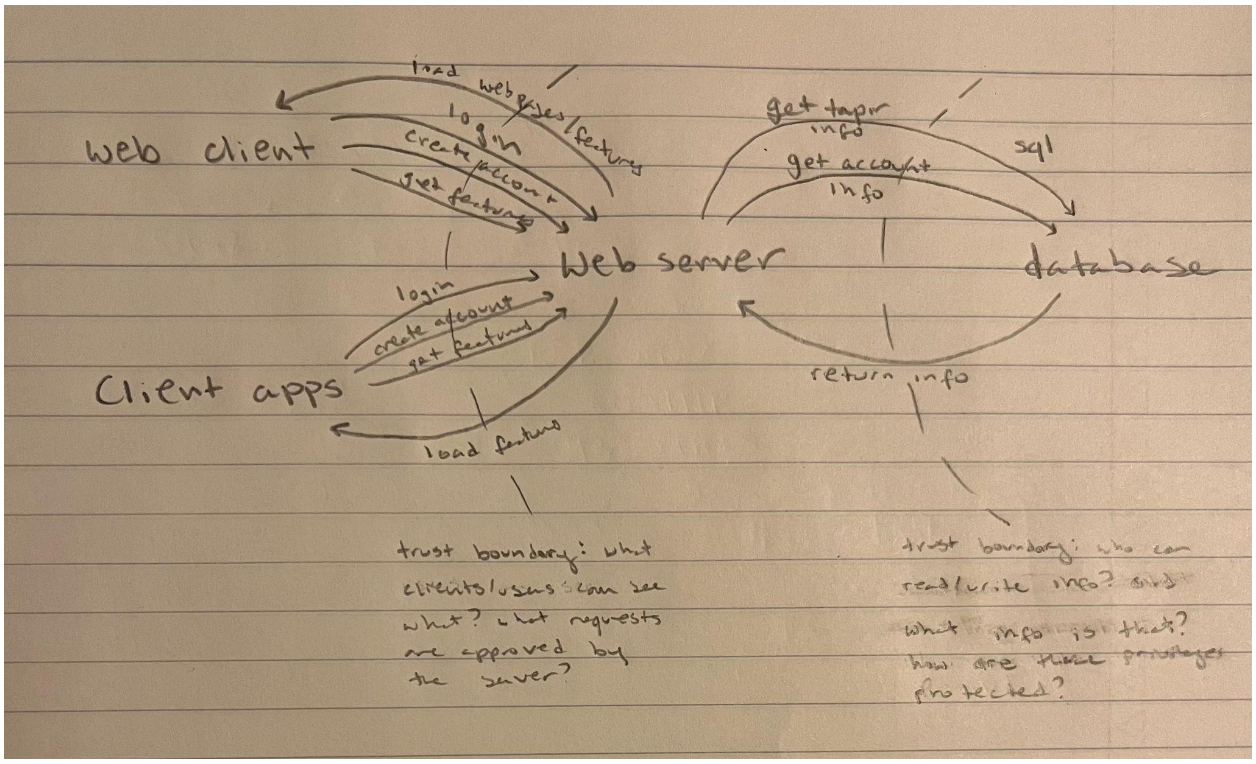
Thea Traw
Threat analysis using STRIDE


web client
load webpages/features
login
create account
get features
Web server
Client apps
login
create account
get features
load features
get tapir info
get account info
sql
database
return info
trust boundary: what clients/users can see what? what requests are approved by the server?
trust boundary: who can read/write info? and what info is that? how are these privileges protected?

Data flow diagram

Threats & mitigations:

Spoofing

- ➢ Eavesdropper acquires the user's username and password during TU account creation or login, and then sends nefarious tapir messages under the user's name.
  - Mitigation: All interactions with the TU server should be encrypted with TLS.
- ➢ A malicious actor breaks into the TU database, steals all of the login info, and then uses it to impersonate various users.
  - Mitigation: Two-Factor authentication could help prevent these logins (or at least without user notification).
  - Mitigation: Store passwords as hashes.
- ➢ A malicious actor could brute-force guess a password for a TU account and then access private information on the TU account (chat logs, etc).
  - Mitigation: Limit the number of login attempts.
  - Mitigation: Two-Factor authentication to (better) confirm the user's identity.

- A fake malicious app is posted, masquerading as the TU client app. A client downloads that app (and its malware) and then any number of bad things happen to the user's device.
  - Mitigation: Make the hash of the actual download public and easy to access (so that users can make sure they are not being tricked).
  - Mitigation: Build-in protection in the App store.

Tampering

- A malicious actor could change all passwords in the TU database and lock everyone out of their accounts. (This results in denial of service but is enacted through tampering.)
  - Mitigation: Only allow specific users to have write-access to specific parts of the database.
- A malicious actor could depossess a user of control of their own TU account by modifying their password during the account creation process. (So the malicious actor knows the password and the user does not–then this would be a stepping stone to a spoofing attack, where the malicious actor could impersonate the user.)
  - Mitigation: Encrypt exchanges with TLS.
- A malicious actor could delete accounts from the TU database, or chat logs, or tapir images, or any number of pieces of information.
  - Mitigation: Only authorize specific users to have write-access over specific parts of the database.
  - Mitigation: Save back-up copies of data.
  - Mitigation: Store different categories of information (say, account info, tapir info, credit card info) on different databases to prevent the range of impact a single attack could have.

Repudiation

- A malicious actor unrightfully gains entry to a user's TU account (potentially through the user not logging out on a shared machine, or the user choosing a weak password, or some other way) and then sends tapir messages and pays for tapir merch while logged into that account. Those actions then cannot be traced back to the actual perpetrator.
  - Mitigation: Track login information (IP addresses, devices) to call attention to irregular behavior and mark it as such.
  - Mitigation: Track logs of user activity to identify patterns of nefarious access (like common scripts used to attack web applications).
- A TU insider uses special administrative access to edit the database and change the chat logs of specific users.
  - Mitigation: There should never be a shared administrative account.
  - Mitigation: Only allow write-access (and probably read-access as well) to specific portions of the database, even for employees.

Information disclosure

- An eavesdropper reads various user information during interaction with the TU server (account creation, credit cards, tapir data).

    - Mitigation: Encrypt exchanges with TLS.
- A malicious actor tries username and password combinations with brute force, and then logs in to read the users' TU chat logs and other private information.
    - Mitigation: Limit number of password attempts.
    - Mitigation: Two-Factor authentication to notify the actual user and help confirm identity.
- A malicious actor uses an SQL injection attack on the TU database that outputs the contents of the database.
    - Mitigation: Never allow the user to directly put unsanitized code into requests to the database.
    - Mitigation: Make use of several databases so that one leak does not expose everything (separate credit cards, tapir info, and account info, for instance).
- A malicious actor tries "forced browsing" and accesses various TU resources that were not intended to be viewable without the necessary authorization.
    - Mitigation: Delete all extraneous files and protect all resources with the proper authorization.

Denial of service

- A DDoS attack is mounted against the TU server.
    - Mitigation: It doesn't seem like there is a clear best way to prevent this attack, but according to this site, outsourcing protection; using a firewall; and just tracking unusual activity are all reasonable steps to take.
- A malicious attacker writes a script to endlessly create new accounts (which fills up the TU database and makes it slower for other users to access the database).
    - Mitigation: Limit account creation attempts from the same device.
- A malicious actor tampers with the TU certificate (and makes it look like it is expired). Many users would leave the website when the warning shows up.
    - Mitigation: Enact proper protection/installation of the certificate.
- A malicious actor tampers with credentials as they are sent to the TU server (changing to an incorrect password, username, format, etc).
    - Mitigation: Implementation of ways to protect the packets from being tampered with. (Using hashes to be able to detect these changes, or TLS to encrypt the messages, is not unhelpful, but they still do not solve the problem of the denial of service issue.)

Escalation of privilege

- A TU insider gives administrative access to the database to an outside entity.
    - Mitigation: Hire less corrupt employees.
- A malicious app or piece of software could be downloaded to a user's device (seemingly unrelated but built to target TU) and then do a buffer overflow attack on the TU app that then modifies its binary. Then the malware could access or commit changes to the database.
    - Mitigation: Restrict to read-only access to the database unless there is proper authentication and authorization.

- ➢ The malware targets the OS of the device and compromises the TLS & HTTPS libraries, such that all of the data passed through the API in the clear is written to a log file before being encrypted. (And this log file is then aided by some other piece of malware to be passed along to the malicious actor.)
    - Mitigation: Keep the OS updated and as best protected from attacks as it can be.
- ➢ A debug mode is left within the TU source code that implements some version of sudo. An attacker figures out how to turn on this debug mode and then uses it to view or change code or the database.
    - Mitigation: Always clean up code before it is introduced to the public.