Jake Martens and Eli Arbogast

CS231 Threat Analysis

May 2, 2021

Spoofing

1. A malevolent actor could create a near duplicate of our website and try to convince users to login using this fake site. One point of access could be through clicking on a link emailed to the users. We can counter this by never emailing links to users and making sure that users know this is our policy.

Tampering

1. Our system is vulnerable to a man-in-the-middle attack between the web server and the database server. For example, a malevolent actor could interrupt the connection and change the data (a picture of a lemur) that the web server submits to the database. We can require that the web server attaches a signature to every message consisting of the encrypted hash digest of the file. The database will have access to the web server's public key and use this to check that the file has not been tampered with.

Repudiation

1. We could encounter a situation where anybody can submit a photo of a lemur with no accountability. We can counter this by requiring user accounts to make a submission.
2. If we are using user accounts, we might find ourselves vulnerable to brute force attacks. We could address this by tracing each failed attempt to log-in to a specific user, making attacks identifiable.

Information disclosure

1. The client application uses an HTTP API to communicate with the web server, which poses an issue if an interloper attempts to eavesdrop on the interaction. We can use an HTTPS-based service instead to encrypt all communications between the application and the server.

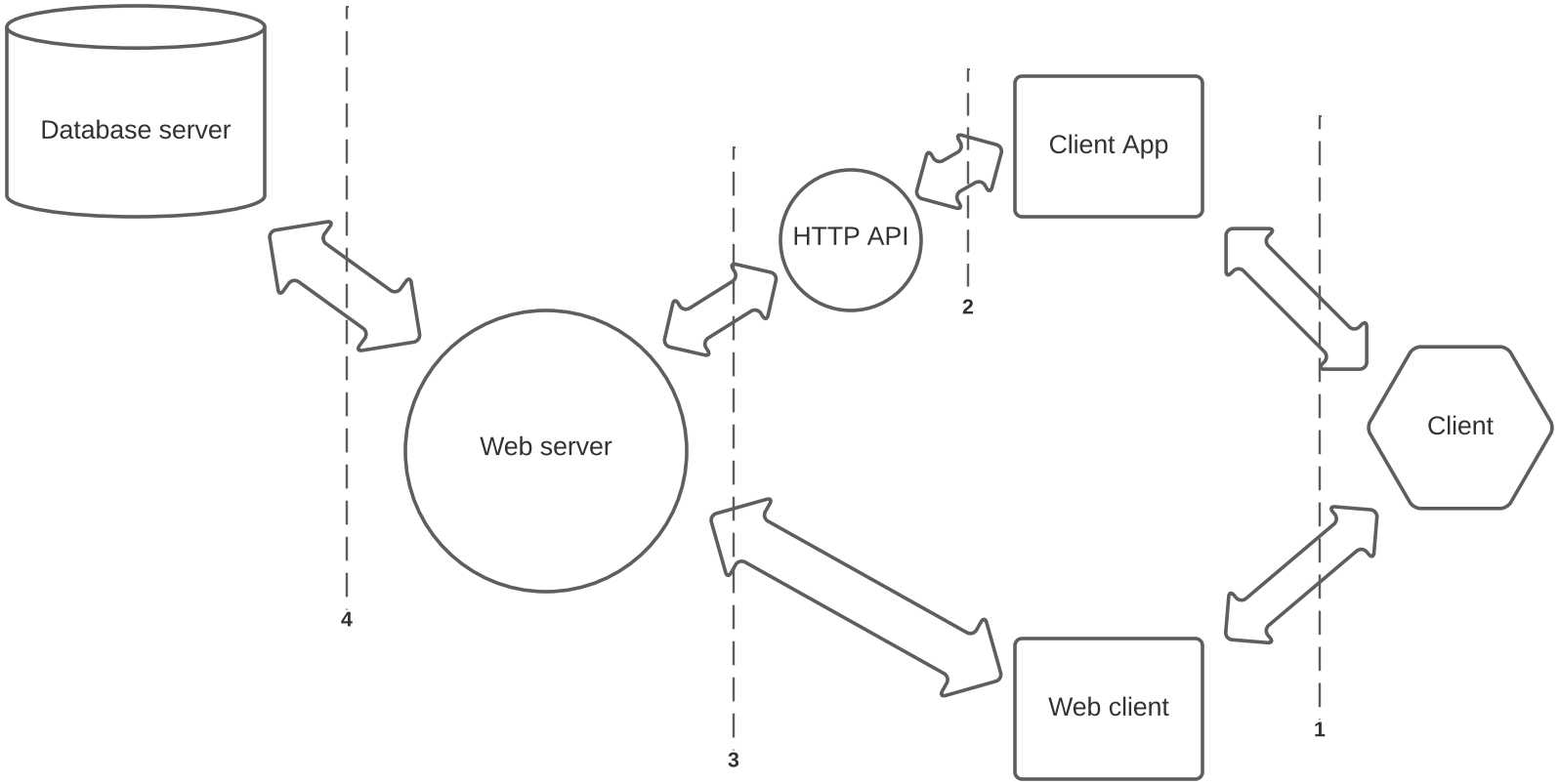
Denial of service

1. One possible attack could be a group signing up for so many accounts or submitting so many photos that our server crashes. We can add a location requirement so that only users in Northfield can sign up. We would somehow verify the address (via IP?) or require approval by a user already a member of the service to join.
2. Another possible attack would be if a malevolent actor had a large selection of bots nefariously attempting to log-in over and over. We could prevent this by having an "I am not a Robot" button on our login page, or employ the services of a company like CloudFlare that can detect and deter spam/bot logins automatically.

Elevation of Privilege

1. We could be vulnerable to someone spoofing a user with elevated privileges on the website, such as a website administrator. To prevent this, we could use digital signatures to make sure that the account that is logged in is indeed from the user linked to it.
2. Our database stores personal information about users that could include addresses and credit card numbers. A malevolent actor could have somehow gained access to a user's account in hopes of accessing this information. We can make this information confidential so that we can never send it from the database to the web server, making it impossible to gain access to it if an account becomes compromised. We can make this

even more secure by hashing this data so that a malevolent actor cannot read this data if they do gain access to the database.


Database server
Client App
HTTP API
2
Web server
Client
4
3
Web client
1