Assignment 6: Security Plan

Brett Bloethner CSIC E-34

User Personas and Stories Overview

User persona and stories (no points, but your assignment will not be accepted without them)

- My primary persona is a 30-year-old software developer named Michael Davis. Michael studied computer science at Stanford and works at Stratos Cloud where he's on a team with 3 other front-end developers who all work together to translate mockups into working code. Most of the mockups come from their design team lead by James Schlotzsky. Michael and his team are constantly having to go back and forth with James and his team regarding missing requirements or just (in their opinion) plain downright poor design. Michael wants to improve communication between his team and the design team and minimize this back and forth to ultimately get more done faster. Being able to communicate better through design assets seems like a good start and Michael is willing to give that a shot with hopes that it may eliminate the long design related threads that make their way into the teams Kanban board software while also minimizing impromptu meetings between his team and the design team. Security is an absolute must for Michael, leaking any design data means competitors could potentially find out what Michael and his company are up to and that's obviously bad news for Michael.

- My secondary persona is a 46-year-old user experience designer names James Schlotzsky. James is on the receiving end of Michaels constant complaining about designs and design requirements. This happens on every project James has worked on though so he's gotten used to it. Regardless, James heard about the application from Michael and his team and is willing to have his group use it as long as it improves communication between everyone and gets the developers to stop bugging him and his design team. James is skeptical about whether the app will really help but he appreciates the effort Michael and his team are putting forward to improve the relationship between the two groups, so he'll give a shot. The most important thing to James is that he and his team are able to communicate the requirements better or at least answer questions quicker and more conveniently, that way their department can meet the tight deadlines set by leadership. James needs the app to be secure as well, the last thing he wants is his teams design documents floating around cyberspace for everyone to copy. At the same time, James needs the application to be low or hassle free. If it means more work than benefit for his team then he'll simply tell Michael it's not the right tool for his team.

Security Requirements

What security requirements apply to users of this application? For example, do they need to authenticate? How often, at what point in the interaction, and with what credentials?

- Users must authenticate before entering the application. Since collaboration between users is the main purpose of the application, some form of identity validation will also be required to ensure users are who they say they are. Standard email address and password authentication is used rather than social auth due to the nature of the application being used in workspaces rather than social settings.

- A one to one relationship between user accounts and email addresses is used to help ensure the identity of users. This combined with email confirmation/validation on signup will give workspace admins the confidence to invite users via email address and rest assured that they invited who they think they invited.

- Deep linking into the application is a vital feature since it enables teams to integrate the app into their agile workflow software by adding links that refer directly to a mockups page. The obvious problem here is that a user shouldn't be hassled with having to authenticate every time they're deep linked into a mockup. To solve this problem, user sessions will be implemented in order to allow users to stay authenticated and bypass login prompts while they're session is active (2 weeks duration after last accessed).

- User sessions will present some security hurdles. If a malicious actor has physical access to a workspace admins browser then they could expose the workspace's design assets and comments via adding an unwanted user to the workspace. They could also delete the workspace entirely. To combat this, an additional successful login (with original email and password) is required for actions that can be especially destructive such as adding a user to a workspace, removing a user from a workspace, deleting a workspace and its data, deleting the user account and changing the user password.

- Provisioning for workspace access is what makes collaboration possible in the app. Workspaces are created by a standard user who then becomes admin of the workspace and has control over all the workspace's settings and assets from an administrative perspective. Once a workspace is deleted, all of the workspace's data is removed. Users gain access to a workspace by accepting an invite from the workspace admin who invited that user via the user's unique email address.

- Applications with similar security requirements and implementations are Slack and Flock and those can be seen as the inspiration behind this applications security design.

Authentication Methods

If they have to obtain authentication credentials, how/when/where will this be done?

- Users obtain their credentials via a signup page by signing up using a unique email address that is not yet associated with other accounts on the platform. After submitting their email address, the user will receive an activation email with a link to create a password and continue into the app. This process ensures that every email address on the platform is unique and also stops bad actors from attempting to register with an email they don't own and locking up that email with a registered but unverified user. Users have no access to any features of the app until they have successfully signed up and are logged into the app.

- Accessing a workspace is done by having the workspace admin add you as a member via your email address. Admins can also add email addresses of non-users and in such cases, a user with that email address will automatically receive an invite to the workspace immediately after signing up on the app. After signing up, users are prompted to either accept pending invites to a workspace or create a workspace of their own.

- Credentials will be required to be re-entered for a few features where the potential for costly or destructive results are high. For example, re-authentication is required to delete an account or a workspace as well as to change a password. Adding and removing workspace members also requires re-authentication since this feature could expose sensitive data to another user. It's important to confirm that the person modifying the workspace is in fact the user administrating the workspace before data is exposed to another user.

- User can reset their password via a standard password reset flow where the user receives a reset password link in the email address linked to their account.

Intrusiveness and Productivity

How intrusive are these requirements on the user? What is the effect on their productivity?

- Authenticating via email address and password shouldn't prove to be much of a hurdle for the user since this is the typical authentication method for the vast majority of web applications and it's likely something that the user is already used to. It's possible that social authentication via Facebook, Twitter, or Google could be easier for the select few that would choose to use social auth for work. However, I would imagine the vast majority of users would rather not connect their social accounts to platforms they use in the workplace.

- The unique email address requirement could be annoying for some users simply because they have to wait for a confirmation email to arrive in their account before they can finish signing up. Due to the signup requirement of a unique and verified email, there will never be a user whose email address is hijacked and already in use. If the user made their account a long time ago and no longer remembers their password, they can always reset their password and continue where they left off.

- Re-authentication is definitely intrusive for users. However, I think many users will understand the security value of re-authenticating for actions like deleting the account and adding members to a workspace. These are also actions that are only likely to be used on a rare occasion, so re-authenticating isn't a hurdle user will have to jump over very often.

- To minimize intrusiveness, user sessions are used so that a user won't have to login on browsers they've logged in with previously. This is a standard feature for many web applications and is implemented in order to minimize the friction for daily users who visit the app often or who may deep link into the app from a link in a separate application.

- Productivity could be most impacted by the user having to wait for a confirmation email to sign up since sometimes emails take a while to arrive. Productivity could also be affected by being asked for login credentials a second time for some actions. However, the number of times a user would be prompted to re-enter their credentials is negligible and nearly zero if the user is not an admin of a workspace. Taking all of that into account, I don't think the security of the app in day-to-day use would be very impactful to the users productivity at all.

Hassle Budget

What is the user's hassle budget, i.e., how much of this crap are they willing to put up with? What is their most likely workaround when it is exceeded? Now that you've thought about this, re-think points 2, 3, and 4.

- This application is designed to be a tool that integrates with the user's day to day workflow in a software development shop or on a software development team. With that in mind, I think that if users find adequate value in the product then they'll be willing to deal with any hassle that's outweighed by an improvement in their workflow.

- Seamless integration is a vital part of staying within the users hassle budget. The idea of storing the user session and allowing for deep linking directly into viewing a mockup helps to minimize this hassle without compromising security too much. If a user needs to login every time they click a link from a virtual Kanban board to view a mockup, then their hassle budget will be exceeded incredibly quick and the app won't be used.

- There could be a small cost in the users hassle budget in regard to signing up with a unique email address or when modifying a workspace and having to login an additional time. However, I anticipate that these occurrences would be pretty rare and users would understand and appreciate the added scrutiny behind validating who's making dangerous changes to an account which may be storing sensitive data of theirs.

- There aren't very many similar applications out there, so I think if a user's hassle budget is fully exhausted then the user will likely try to avoid the application altogether by continuing to collaborate using old methods like inserting mockup image files into project management software and commenting on long threads in that software in addition to setting up face to face meetings to resolve those design problems that will inevitably escalate. The user could also try and repurpose another collaborative tool in order to get similar functionality. Something like google docs could act as a cumbersome alternative since it allows for pasting an image into a doc and having comment threads.

Layers Security Approaches

In what way might a layered approach to security lower the everyday burden on users? For example, Amazon stores credit card numbers in your account to lower the burden of each interaction, hence increasing sales. But if you want merchandise delivered to a new address, or even an existing address that you haven't used for a while, Amazon makes you re-enter your credit card number.

- Like with most security issues, the main concern with the layered approach is in validating the user's identity. We assume the user is who they say they are by allowing them nearly full access when a valid session exists. However, were not 100% confident it actually is that user behind the keyboard so for especially destructive tasks we get further confirmation of the user's identity by prompting for a login again.

- One way to improve the layered approach could be to get more data points in order increase our confidence that the session is valid with that user behind the keyboard. This could be done by doing things like asking the user during login if they're on a public or private machine and raising errors for login attempts from unknown areas or easing security and removing the additional authentication requirement when the app is accessed from a frequently used machine with a frequently used IP in a frequently used city.

Government Regulations

What government or other regulations apply to user security and privacy of this application?

- The application has a pretty narrow use case strictly involved in sharing and collaboration in regard to design assets and I think this helps to reduce its exposure to regulation from the government except in cases where the regulation covers pretty much any web app. Some examples might be…
  - COPPA
    - Special attention would need to be paid in an effort to avoid accidentally collecting information from anyone under 13 years of age
  - Omnibus Crime Control and Safe Streets Act of 1968
    - Where it applies to wiretaps and sharing data of suspected criminals with federal agencies against a "reasonable expectation of privacy"
  - Computer Fraud and Abuse Act of 1984
    - In regard to protection from hacking afford by law to the web application/company

  - Identity Theft Assumption and Deterrence Act of 1998
    - Like any other web application, this application can be used to pose as a different individual if one has access to their email account

- The General Data Protection Regulation will probably be the costliest regulation to limit exposure to since it has actionable requirements for compliance in regard to just about any web application. These requirements are outlined in countless guides online and I've noted a few below that would be obvious requirements.
  - Email marketing must be an opt-in option
  - Privacy policy must describe how the app handles user data and if it's shared
  - Users have a right to be forgotten, so "deleted" data must actually be deleted
  - Users must give consent for the app to use analytics and tracking tools
  - Users must be notified that the web app is using cookies

- Even when considering these regulations, the application still faces exposure to lawsuits since anyone can sue for pretty much anything whether or not it relates to a specific regulation. The best defense is to make sure data is stored securely, that users are notified when their data is about to be shared (with a new workspace member for instance), that data is deleted when the user requests it, that the Privacy Policy and Terms and Conditions are agreed to before signup and that those legal documents adequality minimize the apps exposure to legal trouble.

Thinking Requirement

How much are you requiring the user to think in order to get the security and privacy behavior that is correct for this application? How can you reduce that load?

- I think this application makes the user think a bit more than web apps that may only require a user to login before accessing the full featured application. However, I think this additional tax on the user is warranted by the fact that users of this app are electing to use the app to share data and this data could very well be sensitive to the user and the user's employer.

- Having such a workspace-centric signup flow (being forced to join or create a workspace right after signup) could create some anxiety for users that aren't familiar with the application and how it's based on collaboration between users. This design is already used with many messaging and communication apps but I think it may be something to test and revisit if it starts to show problems.

- The load on the user can always be reduced at the cost of reducing security and that's a fine balance that may need to be adjusted in the future. Given the sharing aspect of the application, I think it's important to prioritize security since a breach early on (whether due to bad design or bad code) could mean an abrupt end to the application. However, one way to minimize the load on a user could be to give the user and option to upload mockups publicly to be accessed via deep linking with a url that contains a token or a lengthy and obscure ID but many would argue that security through obscurity isn't security at all.

- I think presenting agreements and policies in summarized and easy to understand form could also minimize the thinking required by the user. I wouldn't go as far as to modify the TOS and Privacy Policy since I would want that preserved to protect the app. However, I think it would be helpful to summarize the TOS and Privacy Policy in bullets above the document while referring the user to the full document below.