TABoo by the SHAmans: Project Report

Members: Alexander Collado
Kushal Delhiwala
Neil Opena
Manpreet Singh

Project: Browser extension for detecting and reporting Tabnabbing Attacks

CSE 331- Computer Security Fundamentals
Professor Nick Nikiforakis

High-Level Project Details

The project consisted of two main portions: the browser extension and a Python-Flask API + SQL Database

For the Chrome browser extension, we used Chrome APIs, specifically storage and tabs, to detect when users switched tabs and to store screenshots of tabs to detect changes when users switched back. Our extension has a switch to turn on and when it is active, the extension takes a screenshot of a tab if it does not already have a screenshot. When the user exits the tab and re-enters it, the extension will take another screenshot and then compare it to the old screenshot. Our extension has a button for reporting sites that makes a call to a controller in Flask application to add the site to a database of blacklisted sites if that site was not already added.

On the back-end side, we had a Flask application that was linked to our extension. During our research, we discovered how a browser extension could not be directly linked to a database, and there would have to be some sort of backend API which served as a sort of intermediary between the extension and our database. For this, we chose to have a Python Flask application that could be both easily set up and deployed. The Flask application had one data model, called Site, which consisted of the site we wanted to blacklist, a unique ID, and the date/time of addition. We also had two usable controllers, one to add a URL to the blacklist and another to check if a site is blacklisted or safe. These two controllers are called by our extension and are used to manipulate, and retrieve information from our database.

For the database, we used a MySQL relational database hosted on Amazon Web Services (AWS). We also deployed our Flask application on AWS, utilizing the AWS Elastic Beanstalk service.

Distribution Across Team Members

We broke this project up into primary roles: frontend and backend development. Each member of the team had a specific role in the process:

1) Neil
    a) Worked primarily on creating the extension.
    b) Utilized Resemble.js for the tab comparison process.
    c) Developed the logic behind calculating comparisons between two versions of a tab and then highlighting the different regions.
    d) Working on the report and presentation
2) Alex
    a) Worked primarily on the frontend of the extension itself.
    b) This included linking the extension with the backend API and allowing users to report sites they deem as suspicious.
    c) Also worked on a portion of the backend that checked whether or not a site was suspicious.
    d) Working on the report and presentation
3) Kushal
    a) Worked primarily on the backend API.
    b) This included developing a portion of the endpoints we have in the backend.
    c) Developed the models in the backend that are stored in the AWS RDS database.
    d) Working on the report and presentation
4) Manpreet
    a) Worked primarily on the backend API and setting up AWS servers.
    b) Setup, configured and linked the Python-Flask API to the AWS RDS instance that was set up.
    c) Deployed the backend API to AWS Elastic Beanstalk.
    d) Working on the report and presentation.

Instructions To Run Our Extension:

To install the Google Chrome Extension:

1) Clone the Git repo.
2) Under Settings in Google Chrome, go to More Tools > Extensions.
3) Click “Load Unpacked”

4) Select the extension from the clone Git repo; The directory name is “taboo”

Now that the extension is installed, here are the steps to test:

1) Go to a website(for testing purposes, we suggest using https://www.securitee.org/teaching/cse331/projects/project2.html)
2) Make sure that the extension is enabled.
3) A screenshot is taken through the extension which will be used in the comparison process.
4) Make a change to the webpage by manually changing the source code.
5) Go to a different tab, and then come back to the same tab.
6) This will take another screenshot that is used in the next step.
7) This will start the comparison process between the two screenshots that were taken.
8) There will be an alert letting the user know that the comparison process is underway.
9) After some time, the area where a change is made will be highlighted.

Third-Party Content/Code Snippets:

The majority of our third party content for the extension involved using the Resemble.js framework. Below is the GitHub link for that framework:

https://rsmbl.github.io/Resemble.js/

We also utilized portions of a tutorial about Flask and setting up the application from a website *HackersandSlackers*. Below is the link of their tutorial:

https://hackersandslackers.com/manage-database-models-with-flask-sqlalchemy/