


Project / Doc ID #	TASK JEEVES / TS-SEC-ASSIGNMENT
<p style="text-align: center;">SOFTWARE WARRIORS: ASSIGNMENT</p> <div style="text-align: center;">  </div>	
Classification/Distribution	Software Warriors Proprietary, Restricted Distribution
Release/Revision Date	Rev: 4 / 2020-07-03
Project Name	Task Jeeves
Project Summary	Basic task manager web application
Team Members	<p>Darlene Agbayani Julian Kim Craig Opie Joseph Paragas</p> <p>Information and Computer Science University of Hawaii at Manoa https://softwarewarriors.github.io/</p>
Distribution Authorizations, Important Notices, and Notes	
<p><u>This document contains Software Warriors proprietary information. Distribution of this document, or any of the information contained herein, is restricted without written consent from Software Warriors Management.</u></p>	

List of Revisions

Revision	Date	Submitted By	Revision Description
Original	2020-05-29	Craig Opie	Assignment 1
Rev: 1	2020-06-07	Darlene Agbayani	Assignment 2
Rev: 2	2020-06-14	Craig Opie	Assignment 3
Rev: 3	2020-06-21	Craig Opie	Assignment 4
Rev: 4	2020-07-03	Craig Opie	Assignment 5

Table of Contents

List of Revisions	2
Table of Contents	3
Introduction	4
Requirements	5
Security And Privacy Requirements	5
End-User Scenarios	5
Security and Privacy Requirements	5
Track Security Flaws	6
Quality Gates	6
SDL Privacy Bug Bar	6
SDL Security Bug Bar	11
Risk Assessment Plan For Security and Privacy	23
Design	24
Design Requirements	24
Attack Surface Analysis And Reduction	25
Threat Modeling	26
Implementation	27
Approved tools	27
Deprecated/Unsafe Functions	28
Static Analysis	28
Dynamic Analysis	29
Verification	30
Fuzz Testing	30
Attack Surface Review	31
Static Analysis Review	31
Dynamic Analysis Review	31
Release	32
Incident Response Plan	32
Final Security Review	33
Certified Release & Archive Report	34

Installation	35
Running The System	35
Viewing the Running App Locally	36
ESLint	36
Directory Structure	36
Import Conventions	37
Application Functionality	37
Landing Page	37
Login Page	38
Register Page	40
Add Task Page	41
List Task Page	42
Edit Task Page	43
Definition Of Terms	43

Introduction

Our team name is the “Software Warriors”. Our [GitHub Page](#) contains information about each member and our projects. Our first project is “Task Jeeves”, a web application that creates a task list specific to each user. Task Jeeves will require users to authenticate using username and password and then will authorize users to view applicable data that they own. Each task will have a title, details, and due date which can be edited by the user. The user will be able to mark a task as completed to remove the task from the active list. Development tools used for Task Jeeves includes GitHub for version management and agile project management, IntelliJ IDEA using ESLint enhanced to meet the AirBnB coding standards, and will be written in JavaScript.

Our Security Development Lifecycle (SDL) is based on the Microsoft SDL which is an industry-leading software security assurance process. Combining a holistic and practical approach, the SDL introduces security and privacy early and throughout all phases of the development process. The following documentation provides an in-depth description of our SDL methodology and requirements.

Requirements

Security And Privacy Requirements

The content present below defines the security and privacy requirement processes. As well as the processes to be used for keeping track of security flaws.

End-User Scenarios

Usage notes: These scenarios apply to consumers, enterprise clients, and enterprise administrators acting as end users.

Security and Privacy Requirements

Authentication

User creates a new account, an email address will be required. Once the account is created, the user will be prompted to authorize their account using a link sent to the email they provided. If the user does not use the email authorization, access to the application features will be denied.

Password Lockouts

User attempts to login using an incorrect password. This will be allowed up to three times per account. Once the limit is reached, login for that user account will be locked for thirty (30) minutes.

Account Recovery

User attempts to login using incorrect information. They would like to recover their account email or password information. The account recovery will require the user to answer security questions.

Track Security Flaws

Assessment of Risk

Using the Common Vulnerability Scoring System v3.1 (CVSS) risks will be assessed for severity, and then prioritized. These assessments will include the temporal and environmental score data. Based on the severity, risks can be resolved in order of importance.

Tools for Tracking

Bugs are discovered throughout the planning, development, and testing process, a separate Kanban in the GitHub Security Project Repository will be utilized. Bugs will have labels to identify priority during creation and will then be placed into the respective category: Critical, Important, Moderate, Low, or Done.

Quality Gates

The content presented below outlines basic criteria to consider when creating privacy and security processes. It is not an exhaustive list of activities or criteria and should not be treated as such.

SDL Privacy Bug Bar

End-User Scenarios.

Usage notes: These scenarios apply to consumers, enterprise clients, and enterprise administrators acting as end users. For enterprise administrators acting in their administrative role, see the Enterprise Administrators Scenarios.

Critical

Lack of notice and consent

Transfer of sensitive personally identifiable information (PII) from the user's system without prominent notice and explicit opt-in consent in the UI prior to transfer.

Lack of user controls

Ongoing collection and transfer of non-essential PII without the ability within the UI for the user to stop subsequent collection and transfer.

Lack of data protection

PII is collected and stored in a persistent general database without an authentication mechanism for users to access and correct stored PII.

Lack of child protection

Age is not collected for a site or service that is attractive to or directed at children and the site collects, uses, or discloses the user's PII.

Improper use of cookies

Sensitive PII stored in a cookie is not encrypted.

Lack of internal data management and control

Access to PII stored at organization is not restricted only to those who have a valid business need or there is no policy to revoke access after it is no longer required.

Insufficient legal controls

Product or feature transmits data to an agent or independent third party that has not signed a legally approved contract.

Important**Lack of notice and consent**

Transfer of non-sensitive PII from the user's computer without prominent notice and explicit opt-in consent in the UI prior to transfer.

Lack of user controls

Ongoing collection and transfer of non-essential anonymous data without the ability in the UI for the user to stop subsequent collection and transfer.

Lack of data protection

Persistently stored non-sensitive PII lacks a mechanism to prevent unauthorized access. A mechanism is not required where the user is notified in the UI that data will be shared (for example, a folder labeled "Shared").

Data minimization

Sensitive PII transmitted to an independent third party is not necessary to achieve the disclosed business purpose.

Improper use of cookies

Non-sensitive PII stored in a persistent cookie is not encrypted.

Moderate**Lack of user controls**

PII is collected and stored locally as hidden metadata without any means for a user to remove the metadata. PII is accessible by others or may be transmitted if files or folders are shared.

Lack of data protection

Temporarily stored non-sensitive PII lacks a mechanism to prevent unauthorized access during transfer or storage. A mechanism is not required where the sharing of information is obvious (for example, user name) or there is prominent notice.

Data minimization

Non-sensitive PII or anonymous data transmitted to an independent third party is not necessary to achieve disclosed business purposes.

Improper use of cookies

Use of persistent cookie where a session cookie would satisfy the purpose. Or, persisting a cookie for a period that is longer than necessary to satisfy the purpose.

Lack of internal data management and control

Data stored at our organization does not have a retention policy.

Low**Lack of notice and consent**

PII is collected and stored locally as hidden metadata without discoverable notice. PII is not accessible by others and is not transmitted if files or folders are shared.

Enterprise Administration Scenarios.

Usage notes: These scenarios apply to enterprise administrators acting in their administrative role. For Enterprise administrators in an end-user role, see the End User Scenarios.

Critical**Lack of enterprise controls**

Automated data transfer of sensitive PII from the user's system without prominent notice and explicit opt-in consent in the UI from the enterprise administrator prior to transfer.

Insufficient Privacy Disclosure

Deployment or development guide for enterprise administrators provides legal advice.

Important**Lack of enterprise controls**

Automated data transfer of non-sensitive PII or anonymous data from the user's system without prominent notice and explicit opt-in consent in the UI from the enterprise administrators prior to transfer. Notice and consent

must appear in the UI—not through the End-User License Agreement (EULA) or Terms of Service.

Insufficient privacy disclosure

Disclosure to enterprise administrators, such as deployment guide or UX, does not disclose storage or transfer of PII.

Moderate**Lack of enterprise controls**

No mechanism is provided or identified to help the enterprise administrators prevent accidental disclosure of user data (for example, set site permissions).

SDL Security Bug Bar**Server.**

The server bar is usually not appropriate when user interaction is part of the exploitation process. If a Critical vulnerability exists only on server products, and is exploited in a way that requires user interaction and results in the compromise of the server, the severity may be reduced from Critical to Important in accordance with the NEAT/data definition of extensive user interaction presented at the start of the client severity pivot.

Critical

Server summary: Network worms or unavoidable cases where the server is “owned.”

Elevation of privilege: The ability to either execute arbitrary code or obtain more privilege than authorized

Remote anonymous user

Unauthorized file system access: arbitrary writing to the file system

Execution of arbitrary code

SQL injection (that allows code execution)

All write access violations (AV), exploitable read AVs, or integer overflows in remote anonymously callable code

Important

Server summary: Non-default critical scenarios or cases where mitigations exist that can help prevent critical scenarios.

Denial of service: Must be "easy to exploit" by sending a small amount of data or be otherwise quickly induced.

Anonymous

Persistent DoS

- Sending a single malicious TCP packet results in a Blue Screen of Death (BSoD)
- Sending a small number of packets that causes a service failure.

Temporary DoS with amplification

- Sending a small number of packets that causes the system to be unusable for a period of time
- A web server (like IIS) being down for a minute or longer
- A single remote client consuming all available resources (sessions, memory) on a server by establishing sessions and keeping them open.

Authenticated

Persistent DoS against a high value asset

- Sending a small number of packets that causes a service failure for a high value asset in server roles (certificate server, Kerberos server, domain controller), such as when a domain-authenticated user can perform a DoS on a domain controller.

Elevation of privilege: The ability to either execute arbitrary code or to obtain more privilege than intended.

Remote authenticated user

Local authenticated user (Terminal Server)

- Unauthorized file system access: arbitrary writing to the file system
- Execution of arbitrary code

All write AVs, exploitable read AVs, or integer overflows in code that can be accessed by remote or local authenticated users that are not administrators (Administrator scenarios do not have security concerns by definition, but are still reliability issues.)

Information disclosure (targeted)

Cases where the attacker can locate and read information from anywhere on the system, including system information that was not intended or designed to be exposed

- Personally identifiable information (PII) disclosure
 - Disclosure of PII (email addresses, phone numbers, credit card information)

- Attacker can collect PII without user consent or in a covert fashion

Spoofing

An entity (computer, server, user, process) is able to masquerade as a specific entity (user or computer) of his/her choice.

- Web server uses client certificate authentication (SSL) improperly to allow an attacker to be identified as any user of his/her choice
- New protocol is designed to provide remote client authentication, but flaw exists in the protocol that allows a malicious remote user to be seen as a different user of his or her choice

Tampering

Modification of any “high value asset” data in a common or default scenario where the modification persists after restarting the affected software

Permanent or persistent modification of any user or system data used in a common or default scenario

- Modification of application data files or databases in a common or default scenario, such as authenticated SQL injection
- Proxy cache poisoning in a common or default scenario
- Modification of OS or application settings without user consent in a common or default scenario

Security features: Breaking or bypassing any security feature provided.

Note that a vulnerability in a security feature is rated “Important” by default, but the rating may be adjusted based on other considerations as documented in the SDL bug bar.

- Disabling or bypassing a firewall without informing users or gaining consent
- Reconfiguring a firewall and allowing connections to other processes

Moderate

Denial of service

Anonymous

Temporary DoS without amplification in a default/common install.

Multiple remote clients consuming all available resources (sessions, memory) on a server by establishing sessions and keeping them open

Authenticated

Persistent DoS

Logged in Exchange user can send a specific mail message and crash the Exchange Server, and the crash is not due to a write AV, exploitable read AV, or integer overflow

Temporary DoS with amplification in a default/common install

Ordinary SQL Server user executes a stored procedure installed by some product and consumes 100% of the CPU for a few minutes

Information disclosure (targeted)

Cases where the attacker can easily read information on the system from specific locations, including system information, which was not intended/ designed to be exposed.

- Targeted disclosure of anonymous data
- Targeted disclosure of the existence of a file
- Targeted disclosure of a file version number

Spoofing

An entity (computer, server, user, process) is able to masquerade as a different, random entity that cannot be specifically selected.

- Client properly authenticates to server, but server hands back a session from another random user who happens to be connected to the server at the same time

Tampering

Permanent or persistent modification of any user or system data in a specific scenario

- Modification of application data files or databases in a specific scenario
- Proxy cache poisoning in a specific scenario
- Modification of OS/application settings without user consent in a specific scenario

Temporary modification of data in a common or default scenario that does not persist after restarting the OS/application-/session

Security assurances:

A security assurance is either a security feature or another product feature/function that customers expect to offer security protection. Communications have messaged (explicitly or implicitly) that customers can rely on the integrity of the feature, and that's what makes it a security assurance. Security bulletins will be released for a shortcoming in a security assurance that undermines the customer's reliance or trust.

- Processes running with normal “user” privileges cannot gain “admin” privileges unless admin password/credentials have been provided via intentionally authorized methods.
- Internet-based JavaScript running in a browser cannot control anything the host operating system unless the user has explicitly changed the default security settings.

Low**Information disclosure (untargeted)**

Runtime information

- Leak of random heap memory

Tampering

Temporary modification of data in a specific scenario that does not persist after restarting the OS/application

Client.

Extensive user action is defined as:

"User interaction" can only happen in client-driven scenarios.

Normal, simple user actions, like previewing mail, viewing local folders, or file shares, are not extensive user interaction.

"Extensive" includes users manually navigating to a particular website (for example, typing in a URL) or by clicking through a yes/no decision.

"Not extensive" includes users clicking through email links.

NEAT qualifier (applies to warnings only). Demonstrably, the UX is:

Necessary (Does the user really need to be presented with the decision?)

Explained (Does the UX present all the information the user needs to make this decision?)

Actionable (Is there a set of steps users can take to make good decisions in both benign and malicious scenarios?)

Tested (Has the warning been reviewed by multiple people, to make sure people understand how to respond to the warning?)

Clarification: Note that the effect of extensive user interaction is not one level reduction in severity, but is and has been a reduction in severity in certain circumstances where the phrase extensive user interaction appears in the bug bar. The intent is to help customers differentiate fast-spreading and wormable attacks from those, where because the user interacts, the attack is slowed down. This bug bar does not allow you to reduce the Elevation of Privilege below Important because of user interaction.

Critical

Network Worms or unavoidable common browsing/use scenarios where the client is "owned" without warnings or prompts.

Elevation of privilege (remote): The ability to either execute arbitrary code or to obtain more privilege than intended

- Unauthorized file system access: writing to the file system
- Execution of arbitrary code without extensive user action
- All write AVs, exploitable read AVs, stack overflows, or integer overflows in remotely callable code (without extensive user action)

Important

Common browsing/use scenarios where client is “owned” with warnings or prompts, or via extensive actions without prompts. Note that this does not discriminate over the quality/usability of a prompt and likelihood a user might click through the prompt, but just that a prompt of some form exists.

Elevation of privilege (remote)

Execution of arbitrary code with extensive user action

All write AVs, exploitable read AVs, or integer overflows
in remote callable code (with extensive user action)

Elevation of privilege (local)

Local low privilege users can elevate themselves to another user, administrator, or local system.

All write AVs, exploitable read AVs, or integer overflows
in local callable code

Information disclosure (targeted)

Cases where the attacker can locate and read information on the system, including system information that was not intended or designed to be exposed.

Unauthorized file system access: reading from the file
system

Disclosure of PII (email addresses, phone numbers)

Phone home scenarios

Denial of service

System corruption DoS requires re-installation of system and/or components.

Visiting a web page causes registry corruption that makes the machine unbootable

Drive-by DoS

Un-authenticated System DoS

Default exposure

No default security features or boundary mitigations (firewalls)

No user interaction

No audit and punish trail

- Drive-by Bluetooth system DoS or SMS in a mobile phone

Spoofing

Ability for an attacker to present a UI that is different from but visually identical to the UI that users must rely on to make valid trust decisions in a default/common scenario. A trust decision is defined as any time the user takes an action believing some information is being presented by a particular entity—either the system or some specific local or remote source.

Examples:

- Displaying a different URL in the browser's address bar from the URL of the site that the browser is actually displaying in a default/common scenario.

- Displaying a window over the browser's address bar that looks identical to an address bar but displays bogus data in a default/common scenario.
 - Displaying a different file name in a "Do you want to run this program?" dialog box than that of the file that will actually be loaded in a default/common scenario
- Display a "fake" login prompt to gather user or account credentials.

Tampering

Permanent modification of any user data or data used to make trust decisions in a common or default scenario that persists after restarting the OS/application.

Web browser cache poisoning

Modification of significant OS/application settings without user consent

Modification of user data

Security features: Breaking or bypassing any security feature provided

Disabling or bypassing a firewall with informing user or gaining consent

Reconfiguring a firewall and allowing connection to other processes

Using weak encryption or keeping the keys stored in plain text

AccessCheck bypass

Bitlocker bypass; for example not encrypting part of the drive

Syskey bypass, a way to decode the syskey without the password

Moderate**Denial of service**

Permanent DoS requires cold reboot or causes Blue Screen/Bug Check.

Opening a Word document causes the machine to Blue Screen/Bug Check.

Information disclosure (targeted)

Cases where the attacker can read information on the system from known locations, including system information that was not intended or designed to be exposed.

Targeted existence of file

Targeted file version number

Spoofing

Ability for an attacker to present a UI that is different from but visually identical to the UI that users are accustomed to trust in a specific scenario. "Accustomed to trust" is defined as anything a user is commonly familiar with based on normal interaction with the operating system or application but does not typically think of as a "trust decision."

Web browser cache poisoning

Modification of significant OS/application settings without user consent

Modification of user data

Low**Denial of service**

Temporary DoS requires restart of application.

Opening a HTML document causes the browser to crash

Spoofing

Ability for an attacker to present a UI that is different from but visually identical to the UI that is a single part of a bigger attack scenario.

User has to go a “malicious” web site, click on a button in spoofed dialog box, and is then susceptible to a vulnerability based on a different browser bug

Tampering

Temporary modification of any data that does not persist after restarting the OS/application.

Information disclosure (untargeted)

- Leak of random heap memory

Risk Assessment Plan For Security and Privacy

Task Jeeves requires users to sign in to access their private information which transfers private information through the user’s browser to a targeted user base consisting of adults. This is defined as a P1 privacy impact rating by our SDL because we will be storing a user's name, email, password, and private task information. The user gains value from our project by being able to create and update tasks that are accessible from any location by accessing our web application through the world wide web and does not store any information on the user’s device beyond browser session data.

The user will be presented with an alert containing our team's privacy policy during account creation. This policy must be acknowledged and agreed to before the user is able to access any portion of our web application that requires authorization. The same process will occur following any revisions to our privacy policy. Additionally, our privacy policy and public disclosure will be available for review by clicking a link available in the site's footer.

Organizations and users can use our project by creating a task list for productivity by authenticating using a username and password to gain access and authorize the user or organization to view their personal task lists. All PII will be contained in areas requiring authorization, passwords will be encrypted using a hash and salt method or better, and the database will be secured while using regular expressions and prepared statements to prevent unauthorized access through user input. Testing of security and privacy will be performed in accordance with the SDL.

Any incidents involving privacy for this project should be immediately directed to:

Craig Opie

opieca@hawaii.edu

Design

Design Requirements

The intended goal of our project is a web application that creates a task list specific to each user. Therefore our application will have a user-interface with simple accessibility. On the homepage, the user will be given the opportunity to either log-in with an existing account or register for a new account. The homepage also gives a brief description of

what our web-application is about. The user will not be able to create or view any task if they are not logged into their account. If the user were to create a new account, it should be done with a trusted certificate to verify encryption of any data entered by the user.

Once the user has created an account, they are given a profile which they can customize to their liking. The profile page should display all of the tasks that the user has already created. If the user is new to the application or has not created any tasks, there will be a “create task” option for them to create their tasks. The tasks will be organized in a “task stack” which is sorted by time of the most recent creation of the task. For example, if a task was created at 9:30 AM and the user creates another task at 10:00 AM, this task will be placed above the other task. As mentioned in the introduction, each task will have a title, details, and due date which can be edited by the user. The user will be able to mark a task as completed to remove the task from the task stack. Users are able to prioritize their tasks based on their level of importance. For example, if the user has an earlier deadline for a task than another one, they have the ability to click on the task and select the “priority first” option. Priority first tasks are then placed on top of the task stack so that the user will know which tasks should be completed first. When a task is completed, it is marked complete by the user and removed from the task stack and placed in the completed tasks folder. Tasks that are placed in this folder can be placed back onto the task stack if needed. Tasks created should also be done with a certificate to prevent tracking of a user’s data.

Attack Surface Analysis And Reduction

There will be three privilege levels on this web application project: Anonymous, User, and Admin levels. The first level is the **Anonymous Level**. Here, users are able to visit the website without having to log into their account. Mainly, this level is to persuade incoming users to sign-up to use the app. In the **User Level**, users login or create an

account with an email address and password. Users will be able to experience the full-application such as customizing their profile and creating tasks. In the **Admin Level**, the creators of the app are able to modify existing accounts and view messages from users.

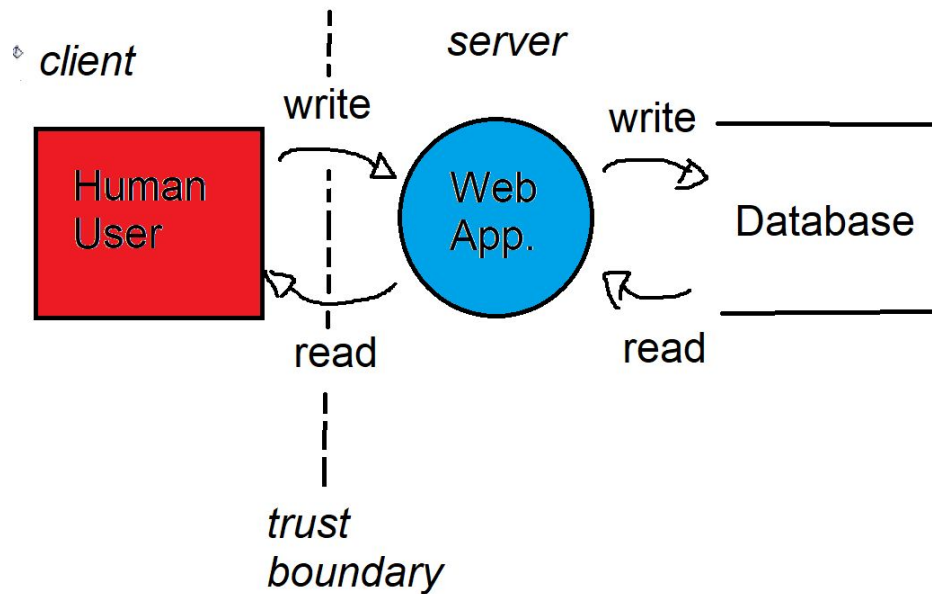
A malicious user may want to exploit the web application to take a person's personal information (email address, username, passwords, daily plans, etc). Listed below are three vulnerabilities web applications could include:

SQL Injections: Attackers can adjust backend SQL statements by manipulating the user provided data

Cross Site Scripting (XSS): Target scripts embedded in a page that are executed on the client side than at the server side

Broken Authentication and Session Management: Websites typically create session cookies and session IDs for each valid session. These cookies contain sensitive data like username, password, etc. If a user is using a public computer and instead of logging off closes the browser. An attacker could use the same system and browse the same website and the user who used the site previously is at risk of having their information stolen.

Threat Modeling



Threats can occur across the privilege boundary between the user input/output and the web application, as data is required to be processed and transmitted between the two, which can lead to vulnerability.

Only account owners should be able to read/write data within their account.

Spoofing the Human User entity

Category: Spoofing

Description: Attackers could potentially falsify data and impersonate a legitimate user to perform malicious acts and gain access to private data

Exposure of sensitive data

Category: Information disclosure

Description: Attackers might gain access to information that should not be available to them like other user's username, password, first/last name, email, location, personal schedule.

Tampering of User Data

Category: Tampering

Description: Attackers with access to an account may be able to alter data maliciously, i.e. add or remove tasks that the owner of the account does not intend.

Implementation

Approved tools

Tool	Version
IntelliJ	11.0.5
Dreamweaver	2019
Windows 10	1909
macOS	10.15.5
Meteor	1.10.2
Semantic UI-React	0.88
React	16.12.0
Semantic UI	2.4.2
Arachni Framework	1.5.1

Deprecated/Unsafe Functions

HTML: *document.write*

Reason: Unsafe function. Writes text directly to the HTML document. Can be used by itself or in conjunction with *document.location.href* and *anyElement.innerHTML* for DOM-based cross-scripting attacks to potentially execute JavaScript on the webpage

Alternative: Use Semantic UI-React components and the text tag to display text

JavaScript: *getYear* and *setYear*

Reason: Deprecated functions. Affected by the “Year-2000-Problem.”

Alternative: *getFullYear* and *setFullYear*

JavaScript: *String.prototype.substr()*

Reason: Deprecated function. Largely obsolete due to the existence of *String.prototype.substring()*. The explanations given in the JavaScript documentation is that “defined in Annex B of the ECMA-262 standard, whose introduction states: ‘... Programmers should not use or assume the existence of these features and behaviours when writing new ECMAScript code. ...’”

Alternative: *String.prototype.substring()*

Static Analysis

Static Analysis Tool	Version
----------------------	---------

ESLint	6.8.0
--------	-------

ESLint is an open source analysis tool for JavaScript. It flags both syntax errors and any code that does not adhere to the set style guideline. ESLint has the advantage of being highly customizable and does not require JavaScript code to be executed to find errors. More information can be found here: <https://eslint.org/docs/about/>.

Dynamic Analysis

We chose to use javascript, react, mongo database, and meteor for our development before knowing about dynamic analysis tools. After extensive research, we decided to use Arachni web application security scanner framework. This decision was largely based on the software being free/public source software, the abundance of security checks that were performed, the highly detailed and well structured reports, and it's availability across operating systems. We deployed Arachni on our web application using the command line version and web application. Using the web application version of Arachni is much more efficient since the report requires Arachni software for viewing the binary.

Our scan revealed that our application is vulnerable to HTTP 'TRACE' method exploits, we are missing the 'X-Frame-Options' header, and our web server discloses the private IP address over the internet. The HTTP 'TRACE' method exploit is evaluated as a moderate threat risk even though the method is not an exploit itself. The method allows a cyber criminal to bypass the HTTPOnly cookie flag which could allow a Cross Site Script (XSS) attack to successfully access a session token. Not including the X-Frame-Options HTTP response header would potentially allow a cyber criminal to render our site inside a frame or iframe exposing our clients to a clickjacking attack and

is evaluated as a low threat risk. A clickjacking attack is a malicious technique where the user is tricked into clicking something different from what they think they are clicking on; furthermore, potentially revealing confidential information or taking control of the user's computer or device.

Verification

Fuzz Testing

Attempt 1: Attempted to input a large amount of characters, 40,000+ (letters, numbers, special characters like >, <, ;, etc.). Into AddTask to test for possible Buffer Overflow, Injection, and Cross-Site Scripting.

Result: Application handles the text properly by converting it to a String in both the name and description of the task. This is due to the application using Meteor and MongoDB Collections and Schemas. The schema specifies the input for the subject and the description of the task to be Strings. This ensures that whatever is inputted by the user is interpreted as a String and has no risk of being executed as possibly malicious code.

Attempt 2: Used SQL Injections on the login page. In the email section, we used an SQL statement (SELECT email, password FROM users WHERE email = "john@foo.com").

Result: When you click the submit button. The application gives you a message saying "A part after @ should not contain the symbol ' ' ". The schema used looks for a specific type of string so in this case, a String as an email address. Therefore from the login angle, there are no vulnerabilities needed to be addressed.

Attempt 3: Used the RockYou plain text password leak as a dictionary for a brute force exploit on our application using python to and the google console to automate the login attempts.

Result: Our passwords for the default admin and user 'john' were discovered after attempt number 3,904. This was a significant issue and so we changed the default passwords on our deployed site to a password that is not contained in the RockYou password list.

Attack Surface Review

As of 6/14/20, there have been no significant changes, updates, or newly reported vulnerabilities to any of the tools in the Approved Tools list in the Implementation section above. Updates will be added to this section as needed.

Static Analysis Review

Our Static Analysis tool of choice is ESLint. As of 6/21/20, running ESLint on our source code resulted in 941 errors. This is due to the ESLint' configuration's strict guidelines on code grammar and style, as the majority of these errors were code-style and grammatical errors, which are trivial to fix. Other errors were related to using "let" instead of "const" when declaring variables, and using html properties as opposed to react properties. As of this report, an issue regarding fixing these ESLint errors is open and currently being worked on in the github project.

Dynamic Analysis Review

Previously, our Arachni scans revealed that our application was vulnerable to HTTP 'TRACE' method exploits, we were missing the 'X-Frame-Options' header, and our web server disclosed the private IP address over the internet. We researched these

issues with meteor and installed the ‘connect’ with ‘webapp’ api and then added the browser-policy package to our meteor application. This allowed us to implement HTTP headers into our server side application which now prevents the exploits. Our scan now reveals that we do not have any security issues on our site.

Release

Incident Response Plan

Privacy Escalation Core Team

- Escalation manager: Julian Kim
 - The Escalation Manager is responsible for overseeing any privacy or security related incident from start to finish. They will also be responsible for including appropriate representation from the team.
- Legal representative: Darlene Agbayani
 - The Legal Representative is responsible for handling the legal concerns that may arise during the escalation handling process
- Public Relations: Joseph Paragas
 - The public relations representative will be responsible for handling any PR concerns that may arise as a result of the security or privacy related incident. They are responsible for writing the official statements released publicly on behalf of the team.
- Security Engineer: Craig Opie
 - The Security Engineer is responsible for leading the process of finding the root cause for the incident within the source code of the application and deploying a technical fix.

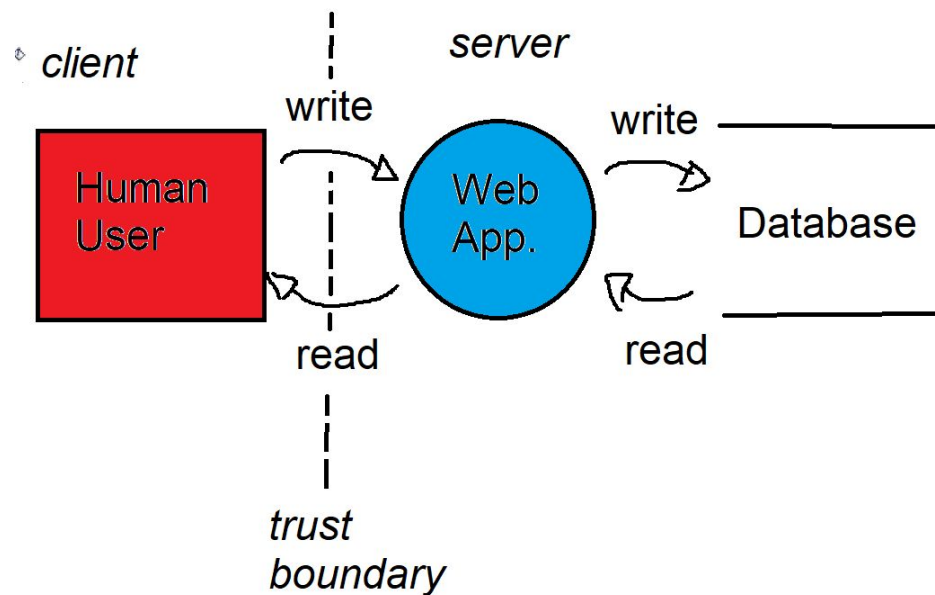
Contact Email: opieca@hawaii.edu

Sample Procedure:

- The Escalation Manager is informed of a security or privacy breach within the application via email contact, etc.
- The Escalation Manager reviews the incident and determines whether there is enough information to solve the issue.
- Information such as:
 - The source of the escalation
 - The impact and breadth of the escalation
 - The validity of the incident or situation
 - A summary of the known facts
 - Timeline expectations
 - Employees who know about the situation, product, or service
- They will then contact the Privacy Escalation team and assign roles and responsibilities
- Each member will handle their responsibilities and once the Escalation Manager ensures that all aspects of the incident have been properly resolved, they will close the issue and review the incident with the team.
- The review process will be to reflect on the incident, the effectiveness of the response, and to ensure that similar incidents do not occur

Final Security Review

Our threat model remained unchanged throughout the development of our SDLC.



All static and dynamic analysis errors have been corrected and our meteor server log indicates that now issues are present. We conducted additional penetration testing on our web application that was limited in scope to not include our web hosting server. After completing our tests and analyzing the results, we settled on a security assessment grade of Passed FSR. Our web application prevents users from creating accounts with the same email address, forces users to use strong passwords, prevents users from inserting javascript into text fields, user passwords are not stored in plain text, HTTP headers are controlled to prevent XSF and clickjacking, and user pages are separated with all private information protected.

Certified Release & Archive Report

The release for Task Jeeves can be found here:

<https://github.com/softwarewarriors/taskjeeves/releases/tag/v1.0.0>

Task Jeeves is at version 1.0.0.

Task Jeeves is a Meteor application that illustrates:

- A secure web application deployment using Microsoft's Security Development Lifecycle
- A task management system that allows users to create tasks for productivity purposes

- Allows creating new tasks with due dates
- Allows marking tasks as complete

Installation

1. Install [Meteor](#).
2. Go to <https://github.com/softwarewarriors/taskjeeves>, and click the “clone or download” button to download your new GitHub repository to your local file system. Using the [GitHub Desktop](#) is a great choice if you use Windows or macOS.
3. Use a terminal application to cd into the app/ directory of your local copy and install third party libraries with the command:

```
meteor npm install
```

Running The System

Once the libraries are installed, you can run the application by invoking the "start" script in the [package.json file](#):

```
meteor npm run start
```

The first time you run the app, it will create some default users and data. Here is the output:

```
=> Started proxy.
=> Started MongoDB.
I20200628-12:12:54.595(-10)? Creating the default user(s)
I20200628-12:12:54.626(-10)?   Creating user admin@foo.com.
I20200628-12:12:54.683(-10)?   Creating user john@foo.com.
I20200628-12:12:54.756(-10)? Creating default tasks.
I20200628-12:12:54.757(-10)?   Adding: First Task (john@foo.com)
I20200628-12:12:54.792(-10)?   Adding: Second Task (john@foo.com)
I20200628-12:12:54.794(-10)?   Adding: Third Task (admin@foo.com)
I20200628-12:12:54.795(-10)?   Adding: Fourth Task (admin@foo.com)
I20200628-12:12:54.797(-10)? Creating default data.
I20200628-12:12:54.797(-10)?   Adding: admin@foo.com
I20200628-12:12:54.831(-10)?   Adding: john@foo.com
=> Started your app.
```

```
=> App running at: http://localhost:3000/
```

Viewing the Running App Locally

If all goes well, the template application will appear at <http://localhost:3000>. You can login using the credentials in [settings.development.json](#), or else register a new account.

ESLint

You can verify that the code obeys our coding standards by running ESLint over the code in the `imports/` directory with:

```
meteor npm run lint
```

Directory Structure

The top-level directory structure is:

```
app/          # holds the Meteor application sources
config/       # holds configuration files, such as
               settings.development.json
doc/          # holds developer documentation, user guides, etc.
.gitignore   # don't commit IntelliJ project files, node_modules, and
               settings.production.json
```

The structure separates documentation files (such as screenshots) and configuration files (such as the settings files) from the actual Meteor application.

The `app/` directory has this structure:

```
client/
  main.html   # The boilerplate HTML with a "root" div to be
               manipulated by React.
  main.js     # Import startup files.
  style.css   # The boilerplate CSS.

imports/
```

```
api/           # Define collections
  task/        # The Task collection definition
  user/        # The User collection definition
startup/       # Define code to run when system starts up
  client/      # Client-only
  server/      # Server-only
ui/           # Define code that details the UI
  components/  # Contains page elements, some of which could
appear on multiple pages.
  layouts/     # Contains top-level layout (<App> component).
  pages/       # Contains components for each page.

node_modules/  # managed by npm

public/        # static assets go here.
  images/      # Contains images available to the public.
  themes/      # Contains themes available.

server/        # Server specific files go here.
  main.js      # Import the server-side js files.
  policy.js    # Import the browser policy js files.
```

Import Conventions

This system adheres to the Meteor guideline of putting all application code in the `imports/` directory, and using `client/main.js` and `server/main.js` to import the code appropriate for the client and server in an appropriate order.

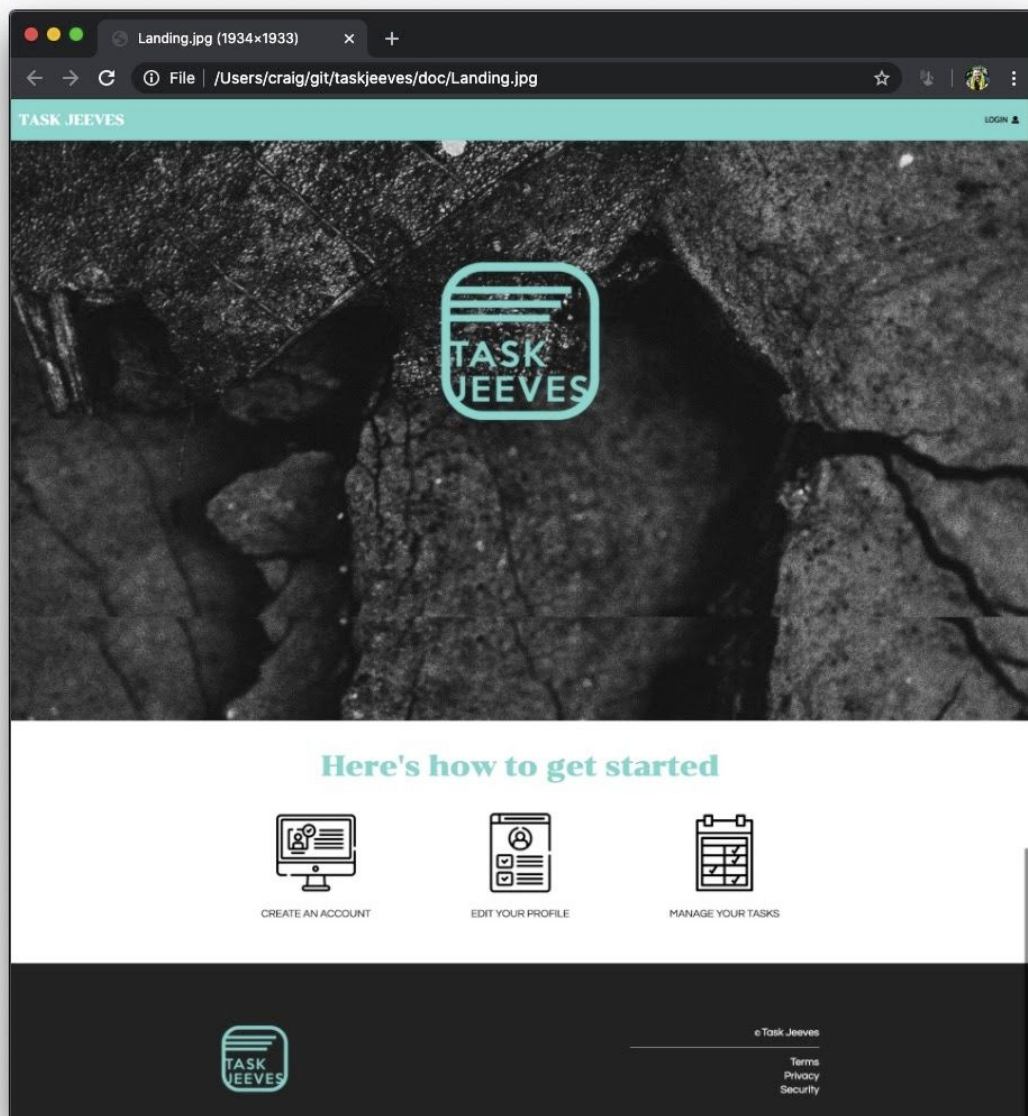
Application Functionality

The application implements a simple CRUD application for managing "Stuff", which is a Mongo Collection consisting of a name (String), a quantity (Number), and a condition (one of 'excellent', 'good', 'fair', or 'poor').

By default, each user only sees the Stuff that they have created. However, the settings file enables you to define default accounts. If you define a user with the role "admin", then that user gets access to a special page which lists all the Stuff defined by all users.

Landing Page

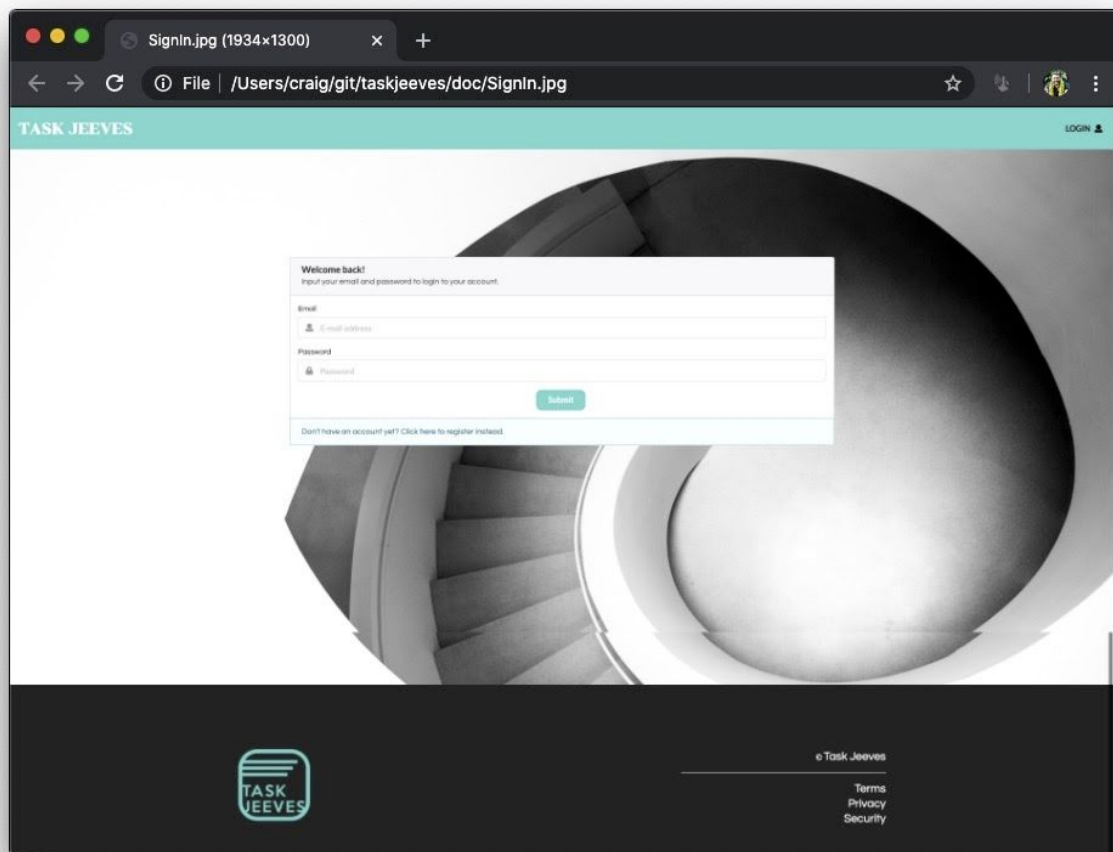
When you retrieve the app at <http://localhost:3000>, this is what should be displayed:



The next step is to use the Login menu to either Login to an existing account or register a new account.

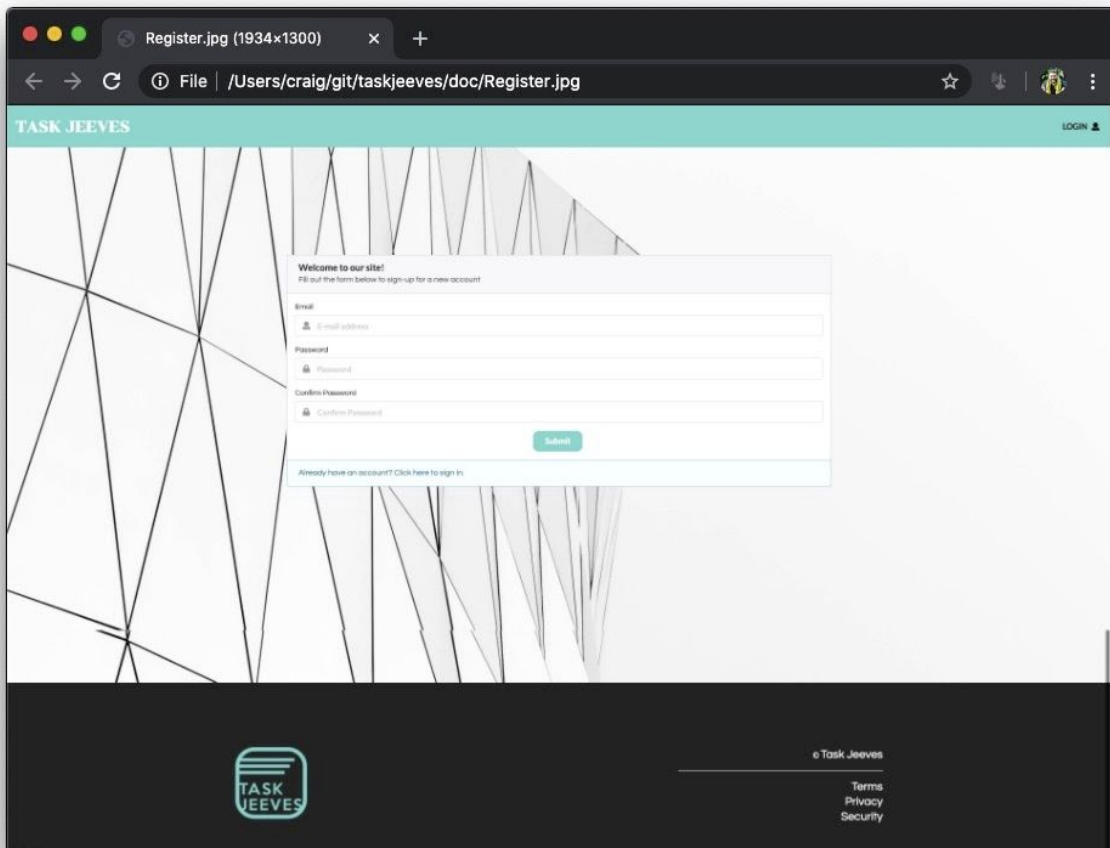
Login Page

Clicking on the Login link, then on the Sign In menu item displays this page:



Register Page

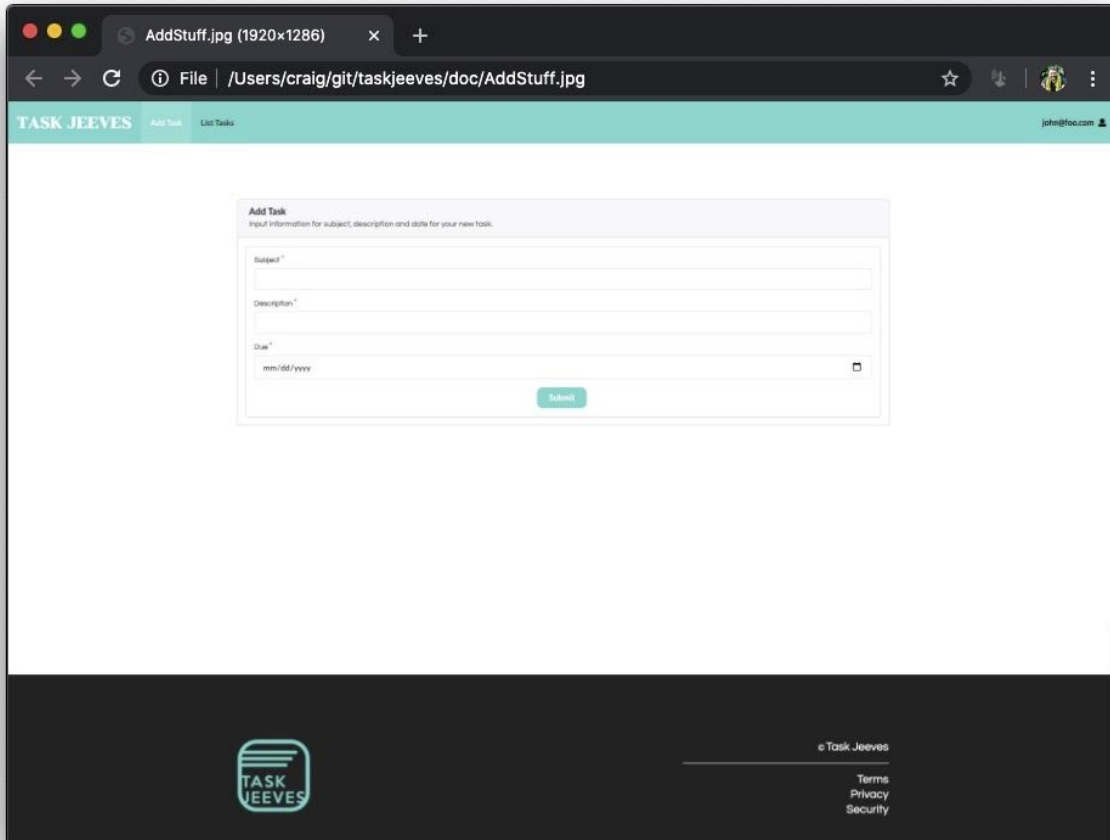
Alternatively, clicking on the Login link, then on the Sign Up menu item displays this page:



The screenshot shows a web browser window displaying the TASK JEEVES registration page. The browser's address bar shows the file path `/Users/craig/git/taskjeeves/doc/Register.jpg`. The page has a teal header with the TASK JEEVES logo and a "LOGIN" link. The main content area features a registration form with the following fields: "Email" (with a sub-label "E-mail address"), "Password", "Confirm Password", and "Confirm Password". A "Submit" button is located below the form. A link "Already have an account? Click here to sign in" is positioned below the form. The footer contains the TASK JEEVES logo, the text "© Task Jeeves", and links for "Terms", "Privacy", and "Security". The background of the page is a light gray with a geometric pattern of black lines.

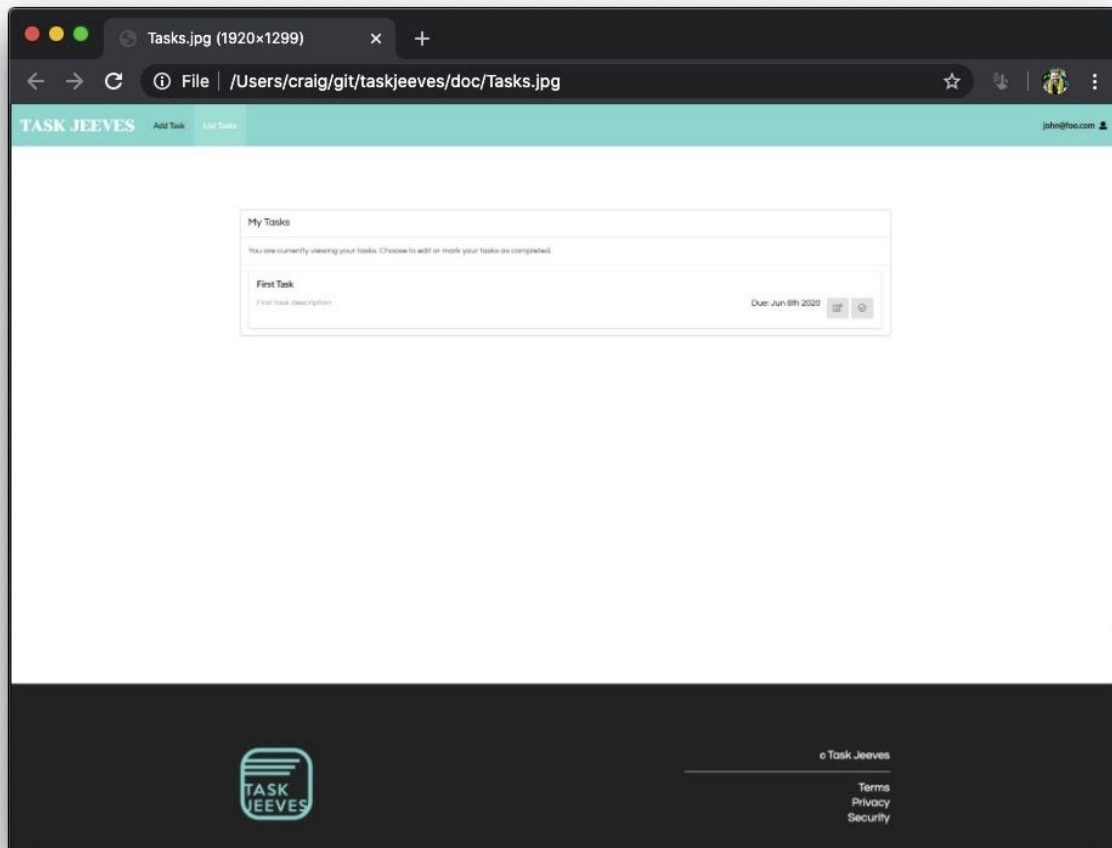
Add Task Page

After logging in, here is the page that allows you to add new Tasks:



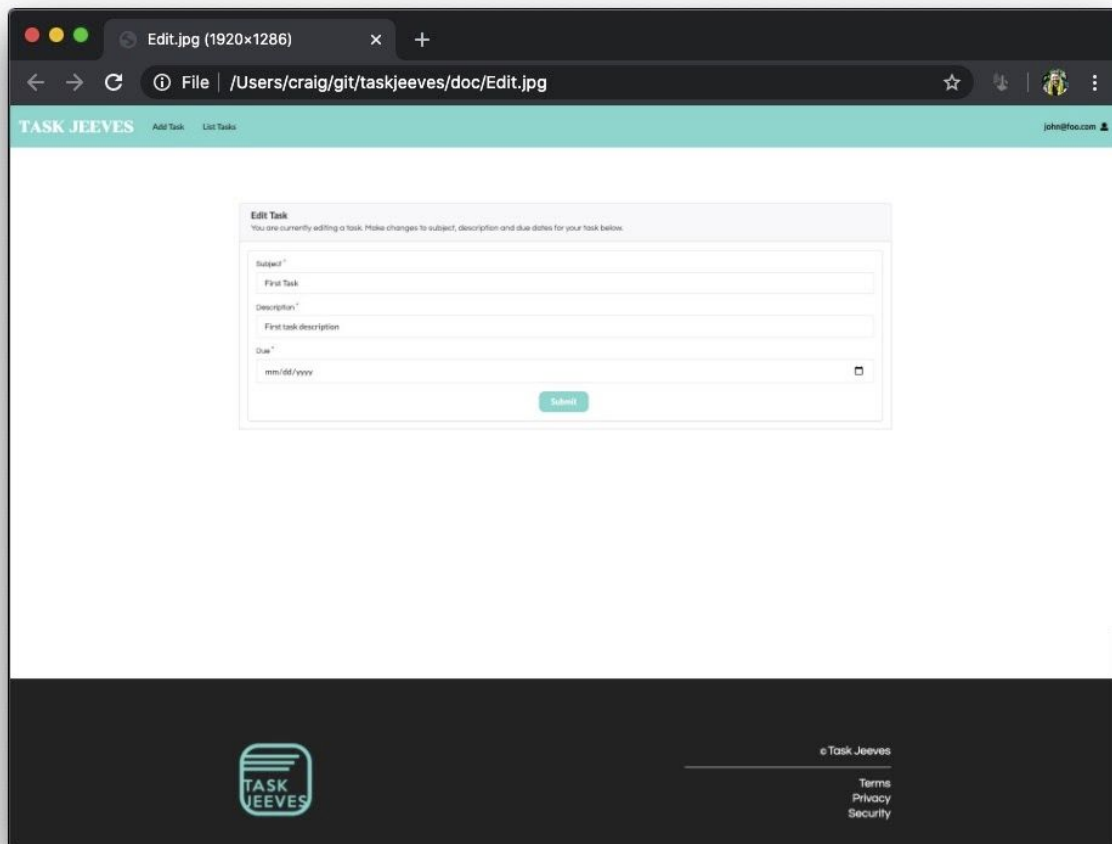
List Task Page

After logging in, here is the page that allows you to list all the Tasks you have created:



Edit Task Page

After clicking on the "Edit" link associated with a task, this page displays that allows you to change and save it:



Definition Of Terms

anonymous data

Non-personal data that has no connection to an individual. By itself, it has no intrinsic link to an individual user. For example, hair color or height (in the absence of other correlating information) does not identify a user.

authenticated

Any attack which has to include authenticating by the network. This implies that logging of some type must be able to occur so that the attacker can be identified.

anonymous

Any attack which does not need to authenticate to complete.

child or children

Under 14 years of age in Korea and under 13 years of age in the United States.

client

Either software that runs locally on a single computer or software that accesses shared resources provided by a server over a network.

default/common

Any features that are active out of the box or that reach more than 10 percent of users.

discoverable notice

A discoverable notice is one the user has to find (for example, by locating and reading a privacy statement of a website or by selecting a privacy statement link from a Help menu).

discrete transfer

Data transfer is discrete when it is an isolated data capture event that is not ongoing.

essential metadata

Metadata that is necessary to the application for supporting the file (for example, file extension).

explicit consent

Explicit consent requires that the user take—or have the ability to take—an explicit action before data is collected or transferred.

hidden metadata

Hidden metadata is information that is stored with a file but is not visible to the user in all views. Hidden data may include personal information or information that the user would likely not want to distribute publicly. If such information is included, the user must be made aware that this information exists and must be given appropriate control over sharing it.

implicit consent

Implicit consent does not require an explicit action indicating consent from the user; the consent is implicit in the operation the user initiates.

non-essential metadata

Metadata that is not necessary to the application for supporting the file (for example, key words).

permanent DoS

A permanent DoS is one that requires an administrator to start, restart, or reinstall all or parts of the system. Any vulnerability that automatically restarts the system is also a permanent DoS.

persistent storage

Persistent storage of data means that the data continues to be available after the user exits the application.

personally identifiable information (PII)

Personally identifiable information is any information (i) that identifies or can be used to identify, contact, or locate the person to whom such information pertains, or (ii) from which identification or contact information of an individual person can be derived. Personally Identifiable Information includes, but is not limited to, name, address, phone number, fax number, e-mail address, financial profiles, medical profile, social security number, and credit card information. Additionally, to the extent that unique information (which by itself is not PII, such as a unique identifier or IP address) is associated with PII, such unique information will also be considered PII.

prominent notice

A prominent notice is one that is designed to catch the user's attention. Prominent notices should contain a high-level, substantive summary of the privacy-impacting aspects of the feature, such as what data is being collected and how that data will be used. The summary should be fully visible to a user without additional action on the part of the user, such as having to scroll down the page. Prominent notices should also include clear instructions for where the user can get additional information (such as in a privacy statement).

scenario

Any features that require special customization or use cases to enable, reaching less than 10 percent of users.

sensitive PII

Sensitive personally identifiable information includes any data that could (i) be used to discriminate (ethnic heritage, religious preference, physical or mental health, for example), (ii) facilitate identity theft (like mother's maiden name), or (iii) permit access to a user's account

(like passwords or PINs). Note that if the data described in this paragraph is not commingled with PII during storage or transfer, and it is not correlated with PII, then the data can be treated as Anonymous Data. If there is any doubt, however, the data should be treated as Sensitive PII. While not technically Sensitive PII, user data that makes users nervous (such as real-time location) should be handled in accordance with the rules for Sensitive PII.

Critical. Release may create legal or regulatory liability for the organization.

Important. Release may create high risk of negative reaction by privacy advocates or damage the organization's image.

Moderate. Some user concerns may be raised, some privacy advocates may question, but repercussions will be limited.

Low. May cause some user queries. Scrutiny by privacy advocates unlikely.

server

Computer that is configured to run software that awaits and fulfills requests from client processes that run on other computers.

Critical. A security vulnerability that would be rated as having the highest potential for damage.

Important. A security vulnerability that would be rated as having significant potential for damage, but less than Critical.

Moderate. A security vulnerability that would be rated as having moderate potential for damage, but less than Important.

Low. A security vulnerability that would be rated as having low potential for damage.

targeted information disclosure

Ability to intentionally select (target) desired information.

temporary DoS

A temporary DoS is a situation where the following criteria are met:

The target cannot perform normal operations due to an attack.

The response to an attack is roughly the same magnitude as the size of the attack.

The target returns to the normal level of functionality shortly after the attack is finished.

The exact definition of "shortly" should be evaluated for each product.

For example, a server is unresponsive while an attacker is constantly sending a stream of packets across a network, and the server returns to normal a few seconds after the packet stream stops.

temporary DoS with amplification

A temporary DoS with amplification is a situation where the following criteria are met:

The target cannot perform normal operations due to an attack.

The response to an attack is magnitudes beyond the size of the attack.

The target returns to the normal level of functionality after the attack is finished, but it takes some time (perhaps a few minutes).

For example, if you can send a malicious 10-byte packet and cause a 2048k response on the network, you are DoSing the bandwidth by amplifying our attack effort.

temporary storage

Temporary storage of data means that the data is only available while the application is running.