Team Beta
Assignment 5

Assignment 5: SDL: Release (Group)
ICS 491

Authors:
Kevin Beydler
Kristin Kogasaka
Sean Baran

Date: 8/6/2016

Abstract: This is the final portion of the SDL project in relation to the Pancake Clicker Application. Essentially, at this point, the project has been fully completed to the best of our abilities and released to the public after performing a few more tests and minor revisions. A comprehensive incident response plan has been established and a link to our GitHub repository is included.

1. Introduction:

It is important to establish security and design requirements early in the Microsoft Security Development Lifecycle at the Requirements and Design Phases respectively to minimize any disruptions to the development schedule [5]. In the Implementation Phase, we consider the most secure method to deploy the project and detect and remove security and privacy issues early on during development of our project [15]. In the Verification Phase, we tested our code to ensure that privacy and security standards are met. Dynamic analysis is performed to detect any issues during run-time. Fuzz testing is conducted to ensure that our project does not contain any security vulnerabilities. In the case of any vulnerabilities, it is necessary to find solutions to make our project ready for secure deployment. The attack surface is reviewed to consider any changes in the design and implementation phases and any new threats that may arise due to these changes [6]. During the Release Phase of the project, our group has created an incident response plan to resolve any future problems and did a final review for security vulnerabilities before finally releasing the project [11].

This report will be going through the Requirements, Design, Implementation, Verification, and Release Phases of the Microsoft SDL for our Pancake Clicker Application.

2. Establish security requirements:

The Pancake Clicker website will hold user information upon registration. This initial user information shall include username, email address, and password. Any logged in user may play the web-based game and the website will keep track of each user’s top score. Of these data fields, email address and password are the only sensitive information collected. All other fields may be viewed by any user, but may not be directly altered by any user.

Each user's password and email address must be kept strictly private and secure, only being accessible to that user. Usernames and email addresses will be non-alterable upon creation of an account. Usernames and email addresses will be unique to each account; no account may share the same username or email address. Usernames and top scores should be viewable to all users. Top scores are only altered through getting a new top score through playing the game. They cannot be altered any other way. Users may only play one game instance at any given time to prevent race conditions in the top score.

Our group will be using GitHub to collaborate and contribute to this project. Everyone in the group is responsible to review all code for possible bugs and vulnerabilities. We will keep track of these bugs and vulnerabilities using the Issues tab on GitHub.

3. Create quality gates/bug bars:

Our group will be using relevant sections of the Microsoft SDL Privacy Bug Bar sample and Microsoft SDL Security Bug Bar sample respectively as follows:

Privacy Bug Bar:

Critical	● Lack of notice and consent Example: 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 data protection Example: PII is collected and stored in a persistent general database without an authentication mechanism for users to access and correct stored PII. ● Improper use of cookies Example: Sensitive PII stored in a cookie is not encrypted. ● Lack of internal data management and control Example: Access to PII stored in the database is not restricted only to those who have a valid business need.
Important	● Lack of notice and consent Example: 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 Example: 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 Example: Persistently stored non-sensitive PII lacks a mechanism to prevent unauthorized access. ● Data minimization Example: Sensitive PII transmitted to an independent third party is not necessary to achieve the disclosed business purpose. ● Improper use of cookies Example: Non-sensitive PII stored in a persistent cookie is not encrypted.
Moderate	● Lack of data protection Example: 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, username) or there is prominent notice. ● Lack of internal data management and control Example: Data stored at organization does not have a retention policy.
Low	● Lack of notice and consent Example: 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.

Note: Adapted from https://msdn.microsoft.com/en-us/library/cc307403.aspx [12]

Server Security Bug Bar:

Critical	Elevation of privilege: The ability to obtain more privilege than authorized ● Remote anonymous user ○ Examples: ■ Unauthorized file system access: arbitrary writing to the file system ■ Execution of arbitrary code ● All write access violations (AV) and exploitable read AVs
Important	● Denial of service: Must be "easy to exploit" by sending a small amount of data or be otherwise quickly induced ○ Persistent DoS ■ Examples: ■ 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
        - Examples:
            - 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
- Elevation of privilege: The ability to obtain more privilege than intended
    - Remote authenticated user
    - Local authenticated user (Terminal Server)
        - Examples:
            - Unauthorized file system access: arbitrary writing to the file system
            - Execution of arbitrary code
    - All write AVs and exploitable read AVs
- 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
        - Example:
            - 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.
        - Examples:
            - 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
    - Permanent or persistent modification of any user or system data used ***in a common or default scenario***
        - Example:
            - Modification of application data files or databases in a common or default scenario, such as authenticated SQL injection

	● Security features: Breaking or bypassing any security feature provided.
Moderate	● Denial of service ○ Temporary DoS ■ Example: ■ Consumption of all available resources on a server by flooding it for a period of time ○ Persistent DoS ■ Example: ■ User can crash the server, and the crash is **not** due to a write AV or exploitable read AV ● 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. ■ Example: ■ Targeted disclosure of anonymous data ■ Targeted disclosure of the existence of a file ● Spoofing ○ An entity (computer, server, user, process) is able to masquerade as a different, random entity that cannot be specifically selected. ■ Example: ■ 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*** ■ Examples: ■ Modification of application data files or databases ***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: ○ Examples: ■ 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 Internet Explorer cannot control anything the host operating system unless the user has explicitly changed the default security settings.

Low	● Tampering ○ Temporary modification of data *in a specific scenario* that does not persist after restarting the OS/application

Note: Adapted from https://msdn.microsoft.com/en-us/library/cc307404.aspx [13]

Client Security Bug Bar:

Critical	Elevation of privilege (remote): The ability to obtain more privilege than intended ● Examples: ○ Unauthorized file system access: writing to the file system ○ All write AVs and exploitable read AVs (***without*** extensive user action)
Important	● Elevation of privilege (remote) ○ All write AVs and exploitable read AVs (***with*** extensive user action) ● Elevation of privilege (local) ○ Local low privilege user can elevate themselves to another user, administrator, or local system. ■ All write AVs and exploitable read AVs 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. ○ Example: ■ Unauthorized file system access: reading from the file system ■ Disclosure of PII (email addresses, phone numbers) ● Denial of service ○ System corruption DoS requires re-installation of system and/or components. ■ Example: ■ Visiting a web page causes registry corruption that makes the machine unbootable ● Spoofing ○ Ability for 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. ■ Examples: ■ Modification of user data ● Security features: Breaking or bypassing any security feature provided ○ Examples: ■ 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
Moderate	● Denial of service ○ Permanent DoS requires cold reboot or causes Blue Screen/Bug Check. ■ Example: ■ Visiting a web page 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. ■ Examples: ■ Targeted existence of file ● Spoofing ○ Ability for 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 application but does not typically think of as a "trust decision." ■ Examples: ■ 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. ■ Example: ■ Opening a HTML document causes Internet Explorer to crash ● Spoofing ○ Ability for 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*. ■ Example: ■ 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.

Note: Adapted from https://msdn.microsoft.com/en-us/library/cc307404.aspx [13]

4. Perform security and privacy risk assessments:

Our project will be called "Pancake Clicker". The public will have access to this project by August 12, 2016. All members of our team shall be responsible for privacy.

We analyzed our privacy impact rating and determined that our website will transfer personally identifiable information (PII) from the user and store it on the server database. Our website also provides an experience that may target children or is attractive to children. Both of these fall under P1 scenarios and require a privacy analysis according to Microsoft's SDL Privacy Questionnaire [10].

The only PII that our website will store are users' email address and password. Notice and consent will be required for the transfer of this PII. This public disclosure will be available when a user attempts to create an account. Email addresses and passwords shall be stored on the database. Passwords will be encrypted in the database for added security. Email addresses shall be viewable only to that user. Passwords will not be viewable to anyone, including the user associated with that password.

Our website will draw users by emphasizing it as a fun, easy, and secure way to pass the time. Users will use the features of the website through the website interface viewable on any modern web browser. Our website and JavaScript game will not install or change anything on any user's computer.

5. Establish design requirements:

Our website will separate server and client files. We will be performing and storing as much as possible on the server side rather than the client side. This is because we do not know what resources are available on the client's device which may affect performance and the server side will be more secure. No sensitive information will be stored on the client side. All user information will be stored and encrypted on the database (MongoDB) on the server side.

The game will feature a pancake that will be clicked in order to receive points, which will be connected to each user's account and displayed via the top scores tab. The clicking will be the sole feature of the game at launch, but there is room for future additions to the gameplay experience.

The top scores tab will show the username and top score for a set number of users. Before displaying the username, we will be sure to "sanitize" it to make sure it does not run any malicious code.

All users will be able to view the homepage containing a brief description of our project and the top scores tab. However, certain parts of the webpage shall only be available to users who have logged in. These parts include the game and user profile. We will check that the user is logged in before displaying these parts.

6. Perform attack surface analysis/reduction:

In our proposed first version of our project, we looked at a similar game known as "Cookie Clicker" [16] to help us determine what types of vulnerabilities users may attempt to exploit on our project. There will be two levels of users - the guest level and the user level. The guest level is restricted and will have very limited access to the features that the website will present, such as only being able to view the top scores and the homepage. Logged in users will have more access to the content available in website, including the game. Both levels of users will be able to view the homepage, which will contain a description of the project, and the top scores page with a number of top user scores. Guest users may register for an account and log-in to attain user access, including access to the game and play to be listed in the top score board.

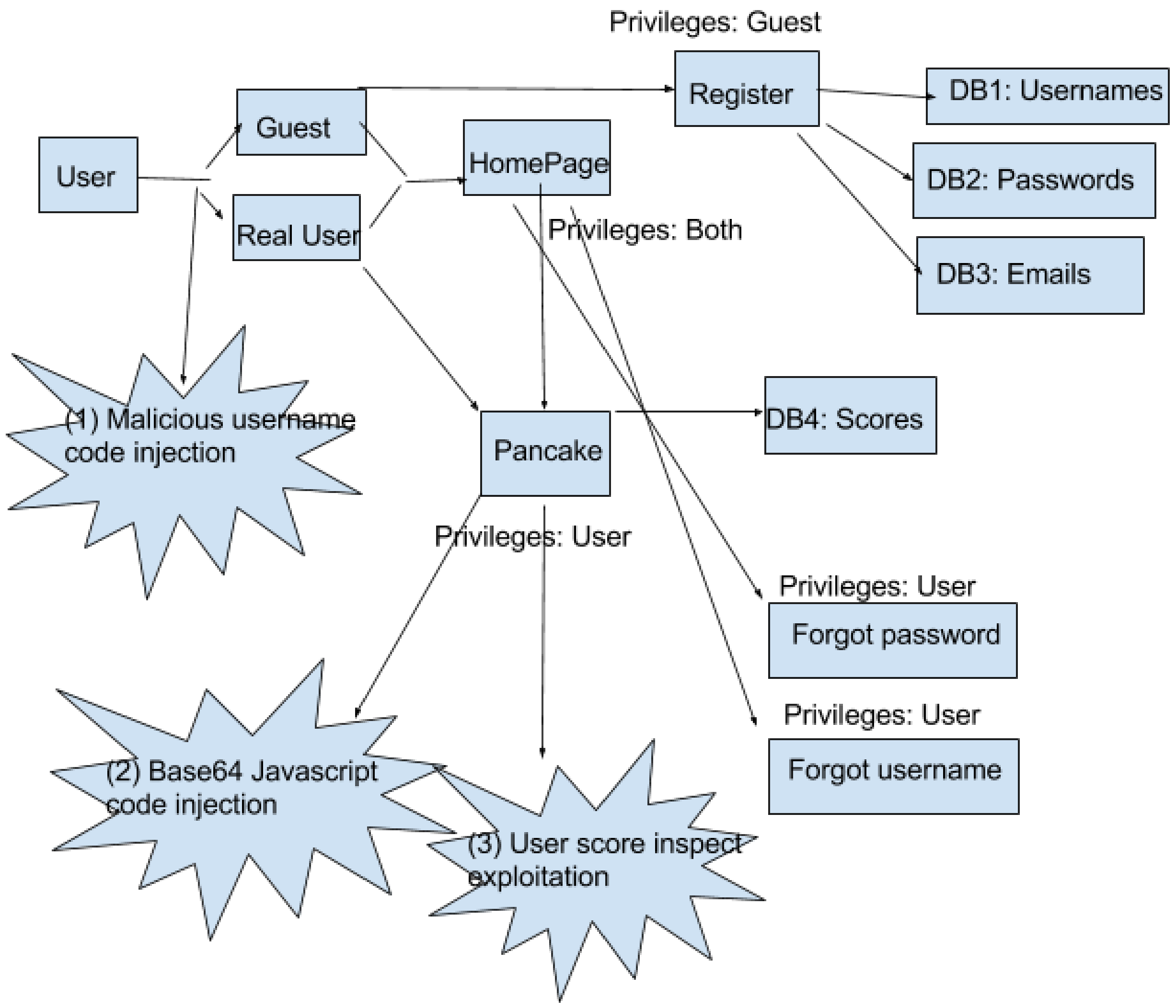
The following is a list of possible vulnerabilities that users may attempt to exploit [14]:

- Attempting to figure out and find/replace segments of the code, then saving the game thereafter (i.e., inspecting the game and then looking for any decimal values associated with the score of the clicks in the game, altering it then saving it as the user's progress).

- Base64 code injection into the possibly vulnerable Javascript code of the project's integrity.
- Malicious username code injection which may run upon displaying the malicious username.

7. Use threat modeling:

Threat Model Diagram:[1]



8. Approved Tools:

The following is a list of approved tools for development, testing, and deployment:

Compiler/Tool	Version	Comments
Meteor	1.3.5.1	JavaScript Application Platform
Intellij IDEA	14.1.4	Any IDE may be used. Our group used Intellij IDEA.
Robomongo	0.8.5	MongoDB manager
Google Chrome Web Browser	51.0.2704.106	Any web browser may be used. This is the browser our group used.
Mozilla Firefox Web Browser	47.0.1	Our group also used this browser testing purposes.
Firebug	2.0.17	Firefox add-on
Command Prompt	10.0.10586	We launched meteor on the local host through the command prompt.

9. <u>Deprecate Unsafe Functions and Meteor Packages:</u>

The following is a comprehensive list of deprecated functions and Meteor packages and safer alternatives to them:

Unsafe or deprecated functions	**Safer alternative**
RegExp compile()	RegExp constructor [9]
valueOf()	Object.valueOf() [7]
Iterator()	for...of loops[8]
Proxy.create and Proxy.createFunction	Proxy [7]
escape and unescape functions	encodeURI, EncodeURIComponent, decodeURI, decodeURIComponent [7]
JavaScript Expression closures	Regular JavaScript expressions [7]
JavaScript String HTML wrapper methods	CSS [7]
zimme:iron-router-active	zimme:active-route [2]
Handlebars.SafeString	Spacebars.SafeString [3]

10. Static Analysis:

The tool we used during the programming of the project was Intellji IDEA's built-in code analyzer tool [4]. What it can do is give us indicators on how our code's integrity and correctness is. We can also run a bulk analysis to analyze the entire code or give a specified analysis run to analyze a smaller segment of code. Impressively, the analyzer tool can also offer what are known as "quick fix" solutions to any of our issues. All of these features are, in a way, similar to the features that the Eclipse IDE offers.

At the time of writing this report, we have used the analyzer several times. Currently, we are only receiving feedback for two things, missing file references, and typos. Both of these are expected as our code is still in the early stages of development, but as our program progresses and we develop more interaction with the database system, it will be important for us to run the tool again to detect more important issues.

11. Perform Dynamic Analysis:

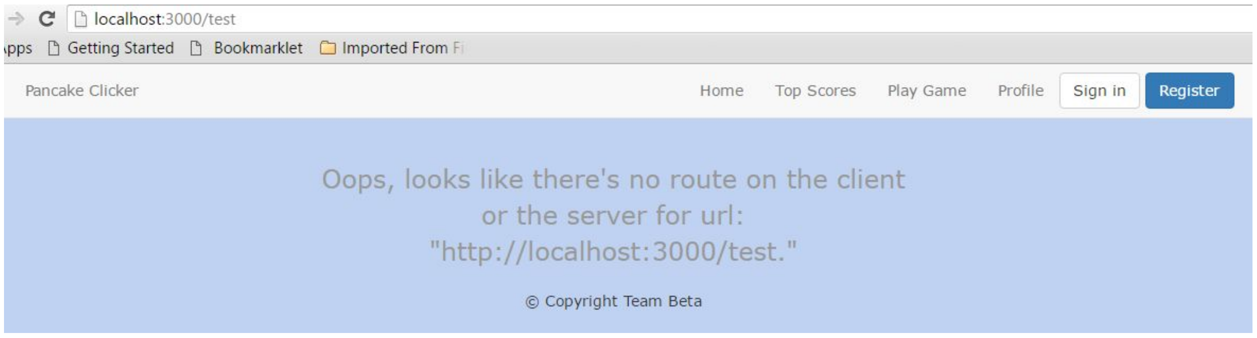
Our group had some difficulties in finding a dynamic analysis tool for JavaScript. We read that performance analysis tools often integrate dynamic analysis techniques [18]. The tool that we have chosen to use is Firefox's Firebug Add-on. We tested each page, watching for any possible errors or warnings. The warning "Password fields present in a form with an insecure (http://) form action. This is a security risk that allows user login credentials to be stolen" did show up in the sign-in page. This is because we are asking the user to input a password in a page that uses http instead of the more secure https. We also noticed the current issue with the save button in the Pancake Clicker game. It currently returns an Object to the console instead of a number. We will be editing this code in the near future.

12. Perform Fuzz Testing :

Our group attempted different techniques to see if we could gain access privileges to pages that users should not have access to, change values of items in the database and items that will be added to the database, and find comments in our HTML and JavaScript code revealing any sensitive information. The following are our findings from these tests.

12.1. Kristin's test:

I created a test.html file to see if it would be accessible even though we had not created any routes to it. The iron-router package prevents users from navigating to any pages that have not been included in the router as shown in Figure 1. Since we do not want this to be the page shown to users, we will be making some adjustments to navigate to a "Not Found" page.

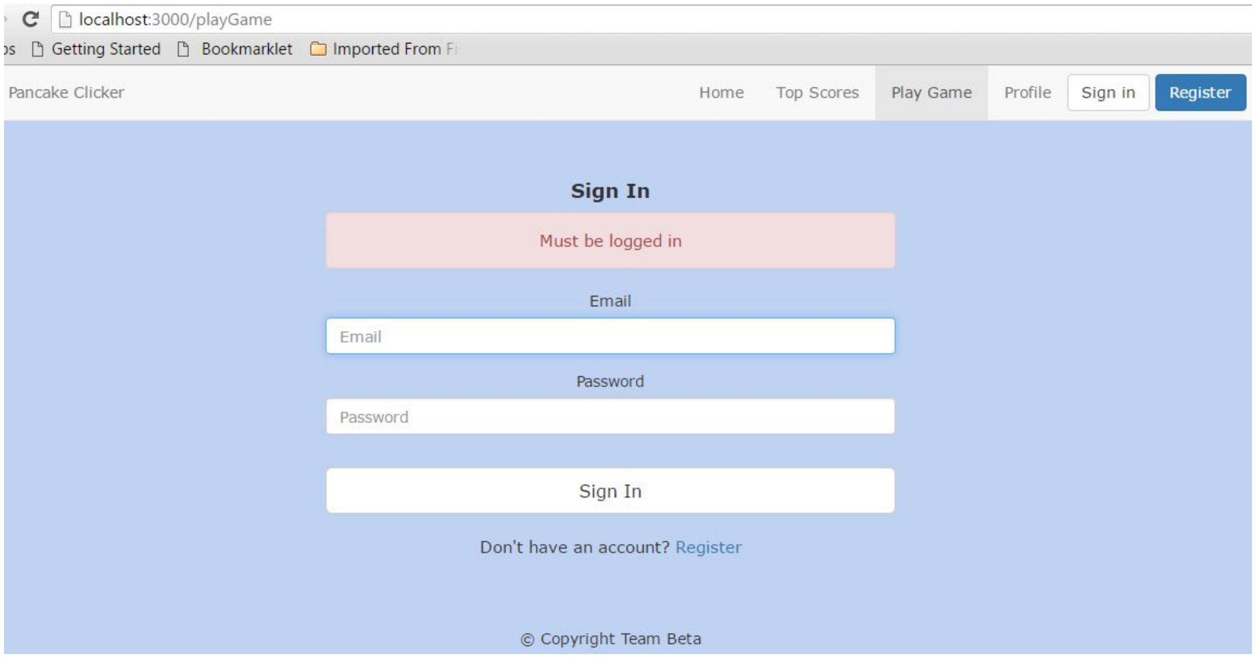


Oops, looks like there's no route on the client or the server for url: "http://localhost:3000/test."

© Copyright Team Beta

Figure 1: Attempt to Navigate to HTML Pages Not in Router

Next, I tried to navigate to our "protected" pages while not logged in. The play game and profile pages are successfully not accessible to guest users as shown in Figure 2.



Sign In

Must be logged in

Email

Password

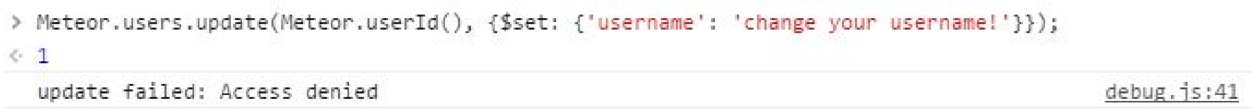
Sign In

Don't have an account? Register

© Copyright Team Beta

Figure 2: Attempt to Navigate to Protected Pages

Next, I did a little research and found an article by David Weldon stating that "User profiles are editable by default even if insecure has been removed" [17]. I decided to test this since it would be bad if users can change whatever fields they want in the database. I decided to log in to a test account and try to change the username. I was denied access in my attempt to change the username as shown in Figure 3.

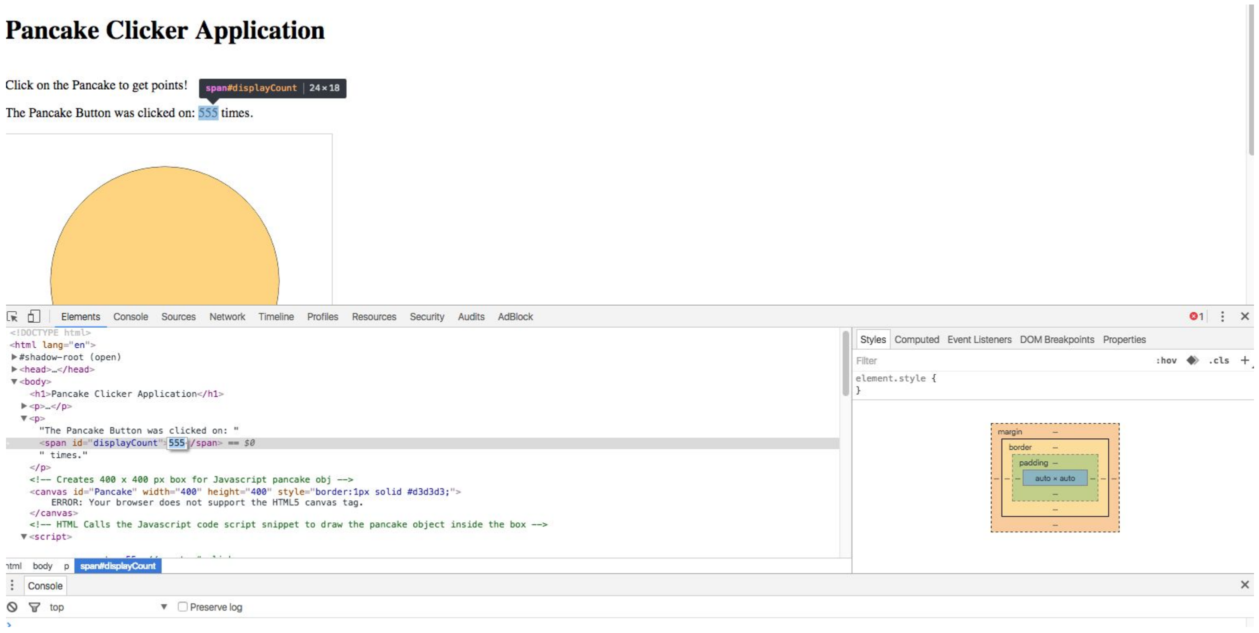


```
> Meteor.users.update(Meteor.userId(), {$set: {'username': 'change your username!'}});
<· 1
  update failed: Access denied                                          debug.js:41
```

Figure 3: Attempt to Change Username

12.2. Kevin Test - Tried to alter the number of clicks made on the user's client side:

It did in fact work and thus, it was indeed considered successful! Using Google Chrome's Inspection tool, I managed to alter the number of clicks by injecting a number that I wanted it to have and it remained on there (Figure 4). Essentially, I can then exit the Inspection page and click the "Save Button" thereafter.



```
<!DOCTYPE html>
<html lang="en">
▶#shadow-root (open)
▶<head>…</head>
▼<body>
   <h1>Pancake Clicker Application</h1>
  ▶<p>…</p>
  ▼<p>
     "The Pancake Button was clicked on: "
     <span id="displayCount">555</span> == $0
     " times."
   </p>
   <!-- Creates 400 x 400 px box for Javascript pancake obj -->
   <canvas id="Pancake" width="400" height="400" style="border:1px solid #d3d3d3;">
       ERROR: Your browser does not support the HTML5 canvas tag.
   </canvas>
   <!-- HTML Calls the Javascript code script snippet to draw the pancake object inside the box -->
  ▼<script>
```

```
element.style {
}
```

Figure 4: Altering local session click count (and then being able to save to the system)

12.3. Sean test - Attempted to search for any Sensitive Data Exposures:

First, I checked to see if we had any secret administrative pages, which we do not because we learned that server configuration pages should never be hidden by a secret non-linked URL. I also searched through our webpages to make sure that all code comments do not reveal any valuable information for attackers such as administrative passwords.

13. Conduct Attack Surface Review:

We had to make a few adjustments to our list of approved tools. We found that older versions of Robomongo did not work with the latest version of Meteor. We have updated our Robomongo to version 0.9.0. We have also included Mozilla Firefox Web browser and the Firebug add-on to conduct dynamic analysis. These change have been made to the list of approved tools in section 8 of this report. These changes have not affected our attack surface since they are tools used for testing.

The only issue our group has found to date is users having the ability to change their score. To fix this issue, we plan to make the counter non-editable, perhaps making the JavaScript react to any changes in the counter.

14. Incident Response Plan:

14.1. Privacy Escalation Team Roles:

1) Kevin Beydler: Public Relations Representative

Kevin will serve at the team's Public Relations Representative who will maintain and sustain the public image of the product and group. Kevin will also adequately represent the Pancake Clicker Application adequately if any Public Relational conflict occurs.

2) Sean Baran: Escalation Manager

The purpose of an escalation manager is to drive the Privacy Escalation Response Framework (PERF) process towards completion and to also appropriately represent the organization. Whenever our team receives an email notification regarding a privacy escalation, the escalation manager will evaluate the content of the escalation and acquire more information if necessary [11].

3) Kristin Kogasaka: Legal Representative

The legal representative shall appropriately represent the Pancake Clicker Application adequately should any legal concern arise. The legal representative shall act on behalf of the organization, making legal decisions in the organization's best interest. This includes, but is not limited to, taking action or hiring outside assistance in the case of an attack or liability issue.

14.2. Group Email Address:

pancakeclicker@gmail.com

14.3. Incident Response Plan [11]:

1) When our group's email account receives an email concerning privacy escalation, the escalation manager shall evaluate the issue and determine whether or not any additional information from the sender is required in order to proceed. The escalation manager shall work with the reporting party and any other associated parties to determine the source of the privacy escalation, the timeline of events related to the incident, the impact of the incident, individuals aware of the incident, and any other relevant information.

2) The escalation manager will then organize and properly document all of the findings inform the appropriate parties.

3) The Escalation Manager will seek a resolution to such an incident by initially assigning portions of the workload to others as needed.

4) The release of appropriate resolutions, such as human resources actions, external communication to the public, proper training, etc., will all be determined by the Privacy Escalation Team as well as the other clients associated with the team during the time of the incident.

5) Upon the complete resolution of the incident, the team will analyze the effectiveness of the Incident Response procedures. This procedure should resolve the concerns of all involved parties and ensure that a similar situation will not arise in the future.

15. Conduct Final Security Review:

After thoroughly testing our web-based project, our group feels that the project passess the final security review with exceptions. Although the website does contain a couple of flaws in the game scoring, no personal information is lost or exposed. One of the flaws in the game scoring are users being able to insert a score through the HTML which will then save to the database upon clicking the "save" button. The other flaw is users receiving points for clicks anywhere within the canvas rather than the pancake image itself.

These flaws do not expose any personal information such as an email address or a password. Users are also not able to gain any unauthorized access and scripts are unable to run from any user input such as the username.

16. Certify Release and Archive:

Pancake Project Link: https://github.com/kfkogasa/PancakeClicker

17. Conclusion:

The Requirements and Design Phases of the Microsoft Security Development Lifecycle, gave our group a better picture of how to go about creating a secure web-based game. The Implementation Phase, helped us continue thinking through how we would go about creating our project before actually jumping into coding. After going through the Verification Phase, we were able to find any security flaws in our project. In the Release Phase, our group created an Incident Response plan to more effectively handle any issues that may arise and though there are still a few flaws in the project, we have tested our project to be safe from exposing any sensitive information.

18. References:

1. Agilemodeling.com. *Data Flow Diagram (DFD)s: An Agile Introduction.* [Online]. Available: http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm

2. Atmospherejs.com. (n.d.). *The trusted source for JavaScript packages, Meteor resources and tools | Atmosphere*. [Online]. Available: https://atmospherejs.com/zimme/iron-router-active

3. Chesters, James. (2014 Apr 8). *Meteor 0.8: Blaze Release Overhauls Rendering System*. [Online]. Available: https://www.infoq.com/news/2014/04/meteor-08-blaze

4. Jetbrains.com. (2016 Apr 20). *Code Analysis.* [Online]. Available: https://www.jetbrains.com/help/idea/2016.1/code-analysis.html

5. Microsoft.com. 2016. *Microsoft Security Development Lifecycle*. [Online]. Available: https://www.microsoft.com/en-us/sdl/default.aspx

6. Microsoft.com. (2016). *Microsoft Security Development Lifecycle*. [Online] Available: https://www.microsoft.com/en-us/SDL/process/verification.aspx

7. Mozilla Developer Network. (2016). *Deprecated and obsolete features*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Deprecated_and_obsolete_features

8. Mozilla Developer Network. (2015). *Iterator*. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Iterator

9. Mozilla Developer Network. (2016). *RegExp.prototype.compile()*. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/compile

10. Msdn.microsoft.com. 2016. *Appendix C: SDL Privacy Questionnaire (Sample).* [Online]. Available: https://msdn.microsoft.com/en-us/library/cc307393.aspx

11. Msdn.microsoft.com. (2016). *Appendix K: SDL Privacy Escalation Response Framework (Sample)*. [Online] Available: https://msdn.microsoft.com/library/cc307401.aspx

12. Msdn.microsoft.com. 2016. *Appendix M: SDL Privacy Bug Bar (Sample)*. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc307403.aspx

13. Msdn.microsoft.com. 2016. *Appendix N: SDL Security Bug Bar (Sample)*. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc307404.aspx

14. Msdn.microsoft.com. 2016. *Fending Off Future Attacks by Reducing Attack Surface.* [Online]. Available: https://msdn.microsoft.com/en-us/library/ms972812.aspx?f=255&MSPPError=-2147217396

15. Msdn.microsoft.com. (2016). *Phase 3: Implementation*. [Online] Available: https://msdn.microsoft.com/en-us/library/cc307416.aspx

16. Orteil.dashnet.org. 2016. *Cookie Clicker.* [Online]. Available: http://orteil.dashnet.org/cookieclicker/

17. Weldon, David. (2014 Dec 2). *David Weldon - meteor: common mistakes*. [Online]. Available: https://dweldon.silvrback.com/common-mistakes

18. Wikipedia. (2016). *Dynamic program analysis*. [Online]. Available: https://en.wikipedia.org/wiki/Dynamic_program_analysis