Team Theta

ICS 491 Assignment 3 SDL: Implementation

Authors: Brian Logan, Brandon Chun, Steven Braun, Jonathan Na

Date: 8/06/2017

Abstract: Since the conception of the web, security has been a priority for all derivations of applications and software products. In modern day, with the advancement of defense/security measures and protocols, hackers have also been evolving their craft in hopes of compromising web applications and other things of the sort for whatever their malicious intentions may be. In this class, we are studying security techniques and protocols that should be in place in order to have a stable and secure web application that users and investors can trust. This assignment shows the steps of assessing security risks and how to deal with them, before any code is ever written.

1. Establish security requirements:

For our application, we felt that the private information listed in the user's account was really the only thing worth protecting. Because of this, we have come up with a few requirements to ensure that information is properly protected.

1. No two users may have the same username and/or email account.
2. Make sure that the user knows how their privacy and security is being kept.
3. Passwords requirement must be no less than 8 characters long [2]
    a. Must include at least one number [2]
    b. Must include at least one uppercase letter [2]
    c. Must include at least one lowercase letter [2]
    d. Must include at least one special character [2]
    e. If a password needs to be changed, the new password cannot be the same as the old password [2].
4. Continuous wrong passwords will lock out the user.
5. An email with a password reset link will be sent to the user if the user forgets his/her password [2].
6. Users cannot see other user's password and personal information
7. Users will only be allowed to edit any information besides key important factors such as the name and username [1].
8. Users are only be allowed to access their own information and gain limited access to the app or program [1].

9. Any breach or attack on the app or program must let the users know and require users to change their passwords if necessary.
10. Must transfer sensitive PII using a secure method that prevents unauthorized access [1].
11. User must be able to control any collection or release of PII [1].
12. Users will be allowed to contact the developers or support for any issues regarding security.
13. The authentication must be able to identify the user [1].
14. There should be no peer to peer contact but only a peer to server and server to peer contact [1].

2. Create quality gates/bug bars

After reviewing the SDL levels of privacy and security, we have chosen the following privacy and security gates/bug bars situations we felt was relevant to our program.

2.1 Privacy

- Lack of notice and consent
- Lack of user control of PII
- Lack of data protection
- Insufficient legal controls
- Improper use of cookies
- Lack of internal data management and control [3]

2.2 Security

2.2a Server Side

- Elevation of privilege: The ability to either execute arbitrary code *or* obtain more privilege than authorized
    - Remote anonymous user
    - All write access violations (AV), exploitable read AVs, or integer overflows in remote anonymously callable code [4]
- 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 [4]
- Spoofing
    - An entity (computer, server, user, process) is able to masquerade as a different, random entity that cannot be specifically selected [4].
- Tampering
    - Permanent or persistent modification of any user or system data ***in a specific scenario***
    - Temporary modification of data in a common or default scenario that does not persist after restarting the OS/application-/session [4].

2.2b Client side

- Elevation of privilege (local)
    - Local low privileged user can elevate themselves to another user, administrator, or local system.

        - All write AVs, exploitable read AVs, or integer overflows in **local** callable code [4]
- 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 [4].
- 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* [4].

3. Perform security and privacy risk assessments

In this section, we will be compiling a list of potential risk assessments that will help us understand the basis of what's needed for program security as well as maintaining the privacy our users would expect from us.

3.1 Identify Your Project and Key Privacy Contacts

- Name of the Project
- Intended purpose
- People responsible for the privacy aspect of the project.

3.2 Privacy Impact Rating

Does the project exhibit any of the following behavior (check all relevant behavior):

___ Stores personally identifiable information (PII) on the user's computer or transfers it from the user's computer [5]

___ Installs new software/extensions or changes file type associations, home page, or search page [5]

___ Transfers anonymous data [5]

___ Runs something silently in the background on a user’s computer

___ Creates any pop-ups

___ None of the above

3.3 Perform a Detailed Privacy Analysis

If you selected anything other than ‘None of the above’ please describe the selected behavior in full detail. (Ex. If your project transfers anonymous data, please describe in full detail how it is done.)

3.4 Test Privacy

Ensure that your program obeys the privacy requirements specified.

3.5 Create Response Team

Establish a team to review the project, in terms of privacy, and if any aspects of the project compromise the integrity of the user’s privacy, escalate it to a privacy advisor.

4. Establish design requirements

User accounts and the private information held within them are essentially the only data worth protecting, so if possible, we would establish the following design:

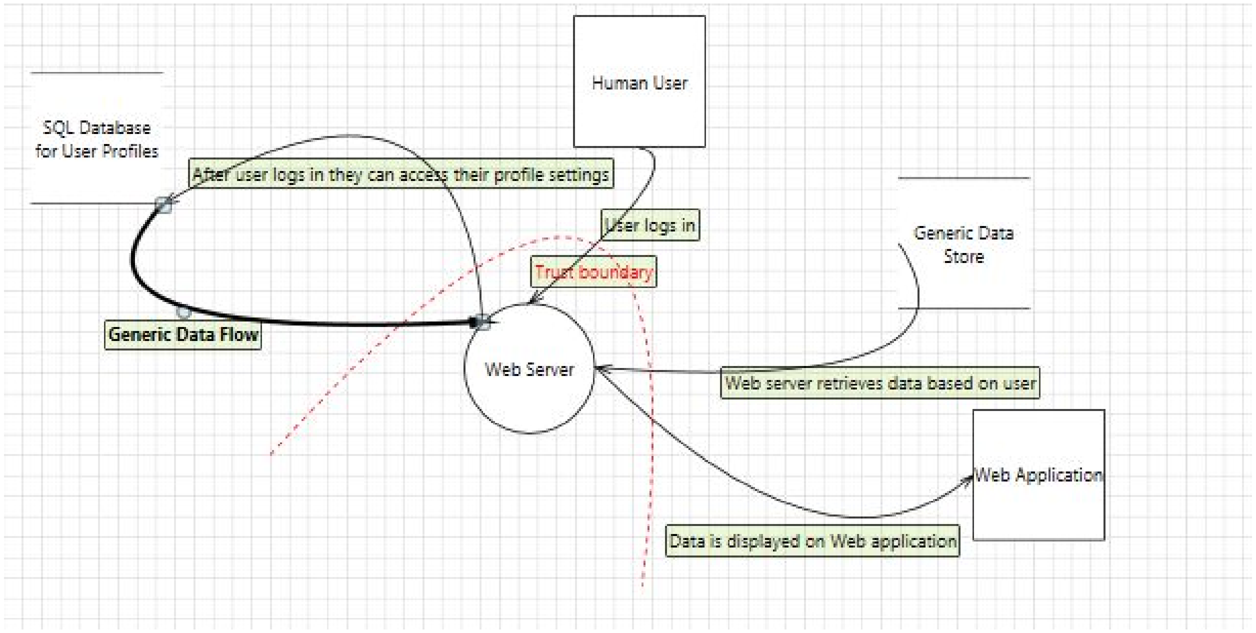
- Require the user to create an account to fully utilize the application.
- When creating a username and password, users will be reminded of the specification required in order for them to properly create credentials.

- Username and password reset/recovery system.
- Some type of multi-factor authentication (optional).

5. Perform attack surface analysis/reduction

- SQL Injections [6]
- Broken Authentication and Session Management (Poor credential management) [6]
- Insecure/weak hashing or encryption of sensitive information [6]
- Weak input validation policies that may enable cross-site scripting [6]
- Insecure Direct Object Reference (If a directory is exposed as a URL or form parameter) [6]
- Security Misconfiguration (Framework, app, web server misconfiguration) [6]
- Non-restrictive URL Access that may allow brute force or scraping attacks [6]
- Insufficient transport protection that doesn't use SSL to encrypt and protect sensitive data translation [6]
- Unsecure website redirects that attackers can implement that reroutes the user from a secure site to a malicious one [6]

6. Use threat modeling


Human User
SQL Database for User Profiles
After user logs in they can access their profile settings
User logs in
Generic Data Store
Trust boundary
Generic Data Flow
Web Server
Web server retrieves data based on user
Web Application
Data is displayed on Web application

Implementation

In this section of the report we have compiled details on our code implementations. The section is divided into 3 subsections which include ‘Approved Tools’, ‘Deprecated/Unsafe Functions’, and ‘Static Analysis’.

1. Approved Tools:

Compiler/Tools	**Version**	**Comments**
Meteor	Version 1.5	The compiler used to compile Javascript and CSS
NPM	Npm-bcrypt 0.9.3 Npm-mongo 2.2.24	
Eclipse Oxygen	Any new version	IDE for Javascript, Java, and CSS
IntelliJ IDEA Ultimate edition	Version 2017.2	IDE for Javascript, Java, and CSS
MongoDb	Version 3.4.6	To allow users to connect to the server

GitHub	Latest version of Git	For sharing and saving files and source code.

2. Deprecated/Unsafe Functions:

- All ‘Meteor.ui’ functions removed. Use ‘Meteor.render’ [7]
- The ‘isAsset’ option to ‘addFiles’ is also deprecated in favor of ‘addAssets’. [7]
- ‘Promise.denodeify’, ‘Promise.nodeify’, ‘Function.prototype.async’, or ‘Function.prototype.asyncApply’, since those APIs have been removed [7]
- The ‘madewith’ package has been removed, as it is not compatible with the new version of that site. [7]
- Remove deprecated ‘Meteor.is_client’ and ‘Meteor.is_server’ variables [7]
- Add ‘dep.depend()’, deprecate ‘Deps.depend(dep)’ and ‘dep.addDependent()’. [7]
- ‘Meteor.uuid()’ is deprecated [7]
- ‘Meteor.render’ and ‘Spark.render’ have been removed. Use ‘UI.render’ and ‘UI.insert instead’. [7]
- The ‘.meteor/cordova-platforms’ file has been renamed to ‘.meteor/platforms’ [7]
- ‘getYear’ and ‘setYear’ are affected by the ‘Year-2000-Problem’ and have been subsumed by ‘getFullYear’ and ‘setFullYear’. [8]
- Deprecate Meteor.autosubscribe. Meteor.subscribe now works within Meteor.autorun. [7]

- ‘meteor-platform’ has been deprecated in favor of the smaller ‘meteor-base’ [7]
- for each...in is deprecated. Use for...of instead. [8]
- Destructuring for...in is deprecated. Use for...of instead. [8]
- let blocks and let expressions are obsolete. [8]

3. Static Analysis:

Since the project will be done using Meteor and Javascript, ESLint will be the code linting(static analysis) utility that we will implement. We did have a slightly difficult time using ESLint but after we successfully configuring it, we were happy with the results it displayed. ESLint allows our group to code within a certain style guideline. This way any code that was written is easily readable and interpreted. Since our code is still a work in progress, we didn’t really have issues regarding problematic patterns and because of that, we couldn’t see the full the extent of what ESLint could do. With that said however, after our experiences with using ESLint, we have come to the conclusion that ESLint is a useful utility that we will continue to use going forth on our project.

Verification

1. **Dynamic Analysis:**

This week we had the decision of choosing between two different types of dynamic analysis tools for our Javascript program. Our choices were either DLint or Jalangi. Unfortunately, we experienced some issues while trying to install both DLint and Jalagi. The first issue that we faced with using DLint is that it would only install on Mac operating systems. We faced this issue when trying to install the dynamic analysis tools on both Linux and Windows computers. For Jalangi our Firefox browsers wouldn't even accept the add-on so we ultimately had to go with DLint instead. However unfortunately, DLint would only work with a Mac OS so it took us a while to get our hands on a working Mac. Apparently even after installing DLint on the Mac we were still unable to get it to work. However due to time limitations we were unable to test any dynamic analysis tools on our project. We also noticed that DLint would only work on public servers and websites however our meteor would have to be deployed in order for DLint to work. If given the proper amount of time and resources (the deployment costs money) we would be able to get our project working on DLint. We will continue to try and get DLint working as we progress with our project.

2. **Fuzz Testing:**

Since our program is based on a checklist, at an attempt to break our program, we've tried adding over 50 different items to the list, seeing if there was an undefined capacity of some sort. This was not the case as the list just kept getting longer and longer as we populated the checklist.

The next thing we tried breaking was the ‘creating and signing in with a username and password’ feature. For creating usernames, it seems like there is no character limit, so username can be as long as the user makes it. Though this isn’t necessarily breaking the program, this is something that was not intended and is something we will be trying to fix.

For our final fuzz testing, we tried to tamper with the inspect element and tried to see if editing anything would give us access to user accounts, or at the very least some information regarding usernames and passwords. This didn’t seem to work as information regarding usernames, passwords, and the items in the checklist in general is stored within mongoDB.

3. Attack Surface Review:

At the time of this review, we’ve searched through all our tools to see if any had been changed and found only a few of our tools had been updated, though nothing too drastic.

Compiler/Tools	**Version**	**Changes Made**
Meteor	Version 1.5	1.5.2 beta was released
NPM	Npm-bcrypt 0.9.3 Npm-mongo 2.2.24	Npm-bcrypt had been updated to 1.0.2
Eclipse Oxygen	Any new version	

IntelliJ IDEA Ultimate edition	Version 2017.2	
MongoDb	Version 3.4.6	No changes
GitHub	Latest version of Git	No changes

Release

1. **Incident Response Plan:**

A privacy escalation team:

Brian Logan - public relations representative - Maintains a positive image of the company in the case of any negative happenings. They are mainly used to craft a good story to tell the media and people. In other words the PR rep is widely used for damage control, such as an attack against a company the PR rep tries to make the company look good and avoid losing the trust of the people. [9]

Brandon Chun - Security engineer - Makes sure that the organization or company is secure and up to date. Pretty much the in house security team that makes sure that the program or website is up to date on firewalls or OS. This role prevents security leaks or attacks from happening. [10]

Steven Braun - Escalation manager - The manager brings the entire team together when a major threat occurs. The manager works with the team until a solid solution has been found. The manager not only works when there isn't a threat but prevents threats as well. For example if a security bug or flaw is found that maybe a potential threat then the escalation manager also brings it to attention to the entire team. [11]

Na, Jonathan - Legal representative - The legal representative should be smart and handsome. The legal rep should represent the organization or company in any legal proceedings. For example if the organization was sued or fined by the government or a customer the legal representative works for the interest of the company by being the representative. [12]

Group email: teamthetahawaii@gmail.com

Procedures in case of an incident:

Purpose: The purpose of this plan is to make a procedure for escalation of attacks against our organization. These procedures also include the privacy escalation team and group email listed above. These procedures are not only for damage control but to also prevent such situations from happening.

Privacy escalation response process: At the first moment of an escalation an emergency email notice will go out to everyone. The escalation manager will be in charge of everything and everyone in the group. First the source of the problem or attack must be found, then determine the impact that the problem or attack and then validate the situation. The manager will also keep track of any information, the timeline and all the employees that are related to the attack. [13] The public relation representative makes prepares a statement to the media and public depending on the situation. If the situation is after an attack then the PR rep makes sure that the damage control is minimum. Any other escalation should be made with an update to the public about the threat and how it was avoided due to the team's efforts. The security engineer makes sure that a patch or fix can be made for the problem or attack. At the same time the engineer should be the one updating the entire team on how critical the attack is and determining the impact of the attack. The dashing legal representative will also closely work with the PR rep to determine any legal repercussions that an attack has been done. What the company's next move is and be the voice of the representative. The PR rep is the writer and the legal rep is the presenter to the public and media.

Closing: Each team member has their own objective to reach the goal of responding to each escalation. Whether big or small each job should not be taken lightly since the customer or clients data may be at risk. After each escalation and incident an updated or patch should be made to prevent future or the same problems from occurring again.

2. Final Security Review :

When conducting the final security review, we found that even though we weren't able to meet all of our intended goals, regarding implementing specific features and restrictions, we were able to at least pass the FSR with exceptions. For brute force attacks, we had intended to increase password enforcement to 8 characters minimum with the inclusion of special character, numbers, and upper and lower case letters. For now we have the password strength set to 6 characters in length with usernames being at least 3 characters long. Though we weren't able to make required specifications to the username and password, we did implement a "timeout" system where too many incorrect attempts would give the attacker a prolonged timeout. We felt this was an acceptable solution in mitigating brute force attacks. We also tried fuzz testing techniques using examples we saw from the Cybrary videos. For sessionID cookies, we realized that Meteor stores session tokens in localStorage after users log in. The Meteor client code reads this token out of localStorage and provides it in a method call to authenticate a DDP connection [14] which ultimately causes CSRF attacks to become futile. For Cross Site Scripting attacks and sql injections, we used similar techniques as for CSRF attacks. Meteor actually uses a page rendering engine that sanitizes input and special characters that protects against very simple XSS and injection attacks. Ultimately, we are not security professionals and know just the basics of security implementation and testing so while it has passed our testing protocols, surely there are more seasoned hackers and techniques that would be able to bypass our security efforts.

GitHub Release: https://github.com/Theta-Team/theta-list/releases/tag/v1.0.0

References

1. First Advantage. (2017). *Access Security Requirements*. [online] Available at: https://www.fadv.com/Fadv-prod/media/Assets/FCRA%20page%20PDFs/AccessSecurityRequirements040809.pdf [Accessed 16 Jul. 2017].
2. Hawaii.edu. (2017). *Change your Password*. [online] Available at: https://www.hawaii.edu/username/userprefs/password_only.cgi [Accessed 16 Jul. 2017].
3. Microsoft. (2017). *Appendix M: SDL Privacy Bug Bar (Sample)*. [online] Available at: https://msdn.microsoft.com/en-us/library/cc307403.aspx
4. Microsoft. (2017). *Appendix N: SDL Security Bug Bar (Sample).* [online] Available at: https://msdn.microsoft.com/en-us/library/cc307404.aspx
5. Msdn.microsoft.com. (2017). *Appendix C: SDL Privacy Questionnaire*. [online] Available at: https://msdn.microsoft.com/en-us/library/cc307393.aspx [Accessed 16 Jul. 2017].
6. eWEEK. (2017). *10 Most Dangerous Web App Security Risks*. [online] Available at: http://www.eweek.com/security/10-most-dangerous-web-app-security-risks [Accessed 16 Jul. 2017].
7. Meteor. (n.d.). *Changelog.* [online] Available at: https://guide.meteor.com/CHANGELOG.html [Accessed 22 Jul. 2017].
8. Mozilla. (2017). *Deprecated and obsolete features*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Deprecated_and_obsolete_features [Accessed 22 Jul. 2017].

9. Public Relations Representative | Inside Jobs. (n.d.). Retrieved August 06, 2017, from http://www.insidejobs.com/careers/public-relations-representative
10. How to Become a Security Engineer | Requirements for Security Engineer Jobs. (n.d.). Retrieved August 06, 2017, from http://www.cyberdegrees.org/jobs/security-engineer/
11. Job Descriptions of an Escalation Manager. (n.d.). Retrieved from http://www.bestsampleresume.com/job-descriptions/manager/escalation-manager.html
12. Legal representative legal definition of legal representative. (n.d.). Retrieved from http://legal-dictionary.thefreedictionary.com/legal+representative
13. Appendix K: SDL Privacy Escalation Response Framework (Sample). (n.d.). Retrieved from https://msdn.microsoft.com/library/cc307401.aspx
14. Group, M. D. (2014, March 14). Why Meteor doesn't use session cookies – Meteor Blog. Retrieved August 07, 2017, from https://blog.meteor.com/why-meteor-doesnt-use-session-cookies-e988544f52c9