**Final Report**

Team 9

Heyt Patel

Wenting Zheng

## Introduction

OpenMRS stands for Open Medical Record System, which is a common framework for creating electronic medical records(EMR). The mission of OpenMRS is to encourage worldwide cooperation to develop and support the electronic medical record management system.

## Observations

### Privacy Policy

Finding the privacy policy is unchallenging since the link is in the bottom banner of every page. Compared to some privacy policies of some software, it is very short and concise but it includes the necessary information that the user should know. The most important thing is highlighted in the privacy policy, “*Please assume that everything you contribute intentionally to OpenMRS is public*”, which alerts the user to pay attention to the personal information they put online especially the identifiable personal information, such as contact and address. Moreover, according to the Code of Fair Information Practices, the privacy policy should include what and how the information will be gathered. In OpenMRS, all the information the user posts on the OpenMRS website, such as emails, code, and documentation, will be public to everyone. Therefore, it’s possible that it will be gathered by some parties. However, the private information will only be disclosed with individuals’ consent unless a valid subpoena is received. In the special case, the individual will be notified of the situation. Lastly, even the Code of Fair Information Practices requires the privacy policy to include a way to correct or modify individual’s identifiable information, it’s not necessary to demonstrate in OpenMRS since most of the information will be public.

### 2. Access Control

There are three access control strategies are used in the OpenMRS, discretionary, role-based access control, and privileges. The discretionary access control is defined as a way to restrict access to an object based on its belongingness. There are two implementation approaches with the owner or with capacities. Approach with the owner means the access will be given by the owner of the object. For example, you will be able to share a Google Docs if you own it. On the other hand, access control with capacities is what is used in OpenMRS. The person who creates the account will be able to add the capacities so that the account can access different views and perform different actions. The administrator should be able to restrict the access to certain files or data by limiting the capacities. The other two strategies are role-based access control, which restricts the access based on roles, and privilege, which determines what the user can do and cannot do in the system. The way it works in OpenMRS is extremely flexible. The role can be used to group privileges and the privileges can also be given separately. What’s more, a role can be a combination of other roles and privileges. This allows combining roles or privileges efficiently and saves space for the special cases. For example, role R1 is defined to contain A, B, and C privileges. If the user needs A B C D, combining role R1 and privilege D will be much easier. If a user just needs B, and C privileges and the combination is really rare in the system, there is no need to create a role to store the role. The user can just be given privileges.

### 3. Password Complexity

OpenMRS allows the user to configure the password complexity in the downloaded software. Since we are using the demo to test, we will only use the default password strength rules. The first rule is the length of the password should be at least eight. When the admin is creating an account, an error message, “At least 8 character(s) are required”, will be given when the entered password doesn’t meet the requirement. The other rule is found in the [wiki](https://wiki.openmrs.org/display/docs/Administering+Users) page. The password should have uppercase, lowercase, and at least one number. The default password complexity is fair. These two rules are applied when the admin creates an account. However, one rule is added when the user is forced to change the password. Here is a screenshot:



The additional rule requires the user to enter the password that doesn’t match the username. It obviously has higher password complexity in this case and leads to a more secure password. In addition, the box of forcing password change is checked by default, which is a good practice of Secure by Default.

### 4.Security by Default

In software security, having a secure network is a fundamental necessity. For strong foundation in software security, the program should select secure configuration by default at the beginning as best foot forward. A secure network is the first verge around software. Therefore, having an unsecured network is an invitation to many vulnerabilities. Open MRS has taken secured step in terms of selecting a network. All the web pages in the system use the HTTPS protocol as shown below. Application of HTTPS sessions provides a secure communication protocol. Also, as shown below it uses the advanced standard security technology like SSL or TSL to ensure that all the data passed between the browser and the web server is encrypted and remains private.

## 

5. Insufficient attack Protection

As we pointed out in phase 1, Open MRS locks out after failed attempts. Having this security prevents attacks done by the brute force method, shoulder suffering, or by guessing patterns over password hint. However, we observe some inaccuracy in terms of approach situation after locking out an account with 7 failed attempts.

For instance, an account is locked out but the error message still shows “Invalid username/password. Please try again.” The error message should show something meaningful. For example, "your account is locked out due to too many failed attempts. Please contact the admin". So, users/ attackers can know that the system is secured against “several failed attempts” an account is locked out. This way, the user with no bad intention knows his/her account is targeted and will let the admin know.

In addition, after several failed attempt, the system blocks an IP address. This can cause a problem to a large number of people. For example, a hospital’s IP address can cause a problem to doctors, lab technician, and many another important user who cannot afford to have blocked the IP address for a minute.

## Vulnerabilities

### Not Fail Securely

When logging in as a user, we try to access the change password page as admin by entering the following URL.

<https://demo.openmrs.org/openmrs/admin/myaccount/changePassword.page>

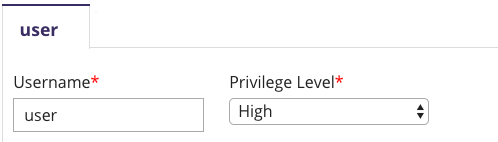
However, an error is thrown with the stack trace. Here is the screenshot of the error:



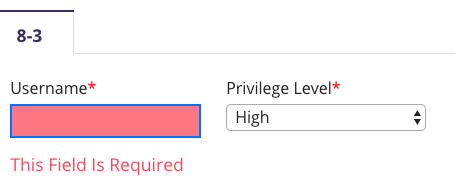
Fail Securely is handling errors securely in such a way that doesn’t give the attacker insight into the implementation of the application. This error message shows the functions the application is using and this may lead to severe consequences. Just like what we did in phase 1 about the vulnerabilities, the attackers could do research on the dependencies to find associated vulnerabilities of certain functions and attack the system.

### 2. Sensitive Data Exposure

Editing the account in the Manage Account Module can result in sensitive data exposure. When we try to edit the USER ACCOUNT DETAILS session, the page should be similar to this screenshot. The username “user” appears twice on the page. One is the label on the top and the other one is the input box.



However, when we leave the input box of Username empty, something strange happens. The label which should be “user” or empty is now a string with some random numbers.



After searching the user in the database, we found out that string is actually the system id of this account. We are not sure where the system ID is used in OpenMRS, but it’s possible that the attacker uses this piece of information for the future attack.



### 3. Broken Access Control

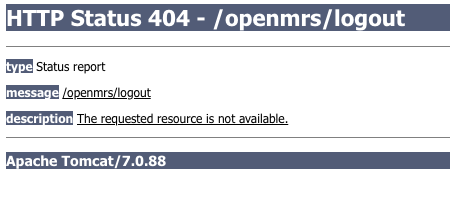
As we discussed in the Access Control session, the capacities are given so that the user can perform different actions and access to certain data in OpenMRS. However, there is a way for the user to access the same data without being given the capacity. For a user who is not given “Uses Patient Summary” capacity, he is not supposed to be able to use “Find Patient Record” feature. However, if he logs in and uses the following URL, the “Find Patient Record” page shows up.

<https://demo.openmrs.org/openmrs/coreapps/findpatient/findPatient.page?app=coreapps.findPatient>

This vulnerability allows the user to bypass the capacity check and obtain unauthorized access to critical data, which results in unwanted modification or exposure of sensitive data.

### 4. Incomplete Logout Implementation

The logout action is allowed when the administrator navigates to the Advanced Administration page. A HTTP 404 error message is given and the administrator is able to access to previous pages



5. Session Management

When a system contains a confidential information, session management should be handle strictly. System is required to log out automatically if the session is inactive for a longer time. Open MRS does not handle session management well enough which does not protect attacker from resuming an abandoned user session. Also, this vulnerability encourages attackers to gain information about current cookie. Since the session does not log out itself cookie does not expire as it supposed to. Because of having accessing to cookie for longer time, attacker will have opportunity to find out password/username. In addition, session does not get invalidated if user closes the browser without logging out .

## Lessons and Suggestions

### 1.Error Message

Many error messages are not secure or user-friendly in OpenMRS. We discussed the insecure error message given in our previous discussion about Not Fail Securely. We learned we should not provide overly informative error messages to the user, but it doesn’t mean simpler error message is better. Error messages that are too simple are not user-friendly. For example, when creating an account, the admin is able to see the error message when the length is less than 8, but no message is given when the password is not the combination of uppercase, lowercase, and the numbers. When we click on the save button, an error message displays, saying “Validation errors found. Failed to save account details.”. This doesn’t provide any meaningful information we can use to correct the “error”. The other example is about the error shown in the login page. According to the [wiki](https://wiki.openmrs.org/display/docs/Administering+Users), the user will be locked out for five minutes after seven failed attempts and the IP address will be locked out after 10 failed attempts. However, after the seven failed attempts, we still get the same message “Invalid username/password. Please try again” with the wrong password. Additionally, we even get the exact same message with the correct password. These are not user-friendly error messages. If the attacker is guessing the password and the account is locked out. The user who really owns the account will receive the message “Invalid username/password” even he enters the correct information. It’s possible that the user thinks he forgot the password and would try the other passwords until his IP address is locked out. In sum, even we should not provide overly informative error messages, some information should be included so that the user can secure their account.

### 2. Defense in Depth

This is a lesson we learned from the Broken Access Control vulnerability in OpenMRS. OpenMRS grants authenticated users the unauthorized access. Authenticated users are the ones whose identities grant access to the system. Authorized users are the ones whose identities grant access to the resource and data. In the example of the Broken Access Control vulnerability, even the capacity is not given, the users can still perform the same action if they know the URL of the page. It’s the same as having all of the capacities including “Administers System”, which have the power to manage the application. Obtaining unauthorized access gives authenticated users abilities to view and modify the critical data. It may result in sensitive data exposure and unauthorized modification of data. Defense in Depth is good solution to this vulnerability. Defense in Depth is using multiple security checks in the system to ensure information security. It can be physical or technical. OpenMRS will focus on technical approach to secure the sensitive data in this case.

### 3. Secure by Default

The default settings are the most secure in OpenMRS. One example is when the admin creates a new account. The Force Password Change box is checked by default. As we discussed previously, it’s the most secure choice for the following two reasons. First, the password rules are more complex when the user changes the password, which means higher security. Then, the user will be the only one who knows the password of his own account. The other secure default setting is the capacity. All boxes of the capacities are unchecked by default. This is the most secure setting if the newly created user doesn’t have access to any data. This obeys the principle of least privilege by unchecking all the boxes initially.

4. More Code More Trouble

As lines of code are increases, the responsibility of sanitization and maintenance also increases. The software may be proved secure in current time, but there is no guarantee to be secure always as the attacker are finding new vulnerabilities every day. Therefore, Developer may find a hard time to maintain and mitigate attacks as the code is extremely long and hard to find a substitute in a small amount of time.

5. Vulnerabilities via Dependency

Research says that “up to 90 percent of applications typically consist of third-party components and more than 50 percent of the world’s largest corporations have open source application with security vulnerabilities”. The previous sentence proves that the developer should not trust external sources or 3rd party APIS when it comes to software security. However, it does not mean that developers are not allowed to use third-party APIs or dependencies. Developers need to be conscious of the vulnerabilities and other important components about dependencies and third-party APIs before using them in their implementation. Also, the developer should learn about the security objective of their product and how those vulnerable dependencies will hurt the organization and their own product. As we mentioned in phase 1, Open MRS also uses many vulnerable dependencies and third party resources. In addition, used dependencies in Open MRS had major problems like unauthorized disclosure of information, unauthorized modification, and disruption of service (Velocity –tools(2.0)). In conclusion, identify the risks from open source libraries early can reduce unplanned work.

## Suggestions

### 1.Error Message

In the previous session, we discussed the impact of the simple error messages by lockout example. In this example, our suggestion is to change the message to “Your account is locked out because of too many failed attempts. Please try after 5 minutes.”when the user enters the correct username and password. In this way, the user who owns the account knows his account is targeted and is able to secure the account by changing the current simple password to a complex one or reporting to the administrator.

Also, it does not fail securely as we discussed in implementation-level vulnerabilities . When you try to redirect the links, it throws an internal server error with stack trace. Stack trace contains information about classes used in software. This can be mitigated by handling internal server error by replace with custom error message.

### 2. Defense in Depth

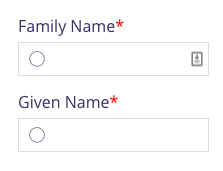
The definition of defense in depth is given in lessons session, so we will include some specific suggestions in this part. Since there are three access control strategies integrated into the system, there should be at least three security checks before the user actually reaches the page. The first check is the authentication. The user should be authenticated to the system to be able to perform any actions. Multi-factor authentication is an excellent method to confirm the user’s identity. The second check is the capacity check. The user has to obtain the capacity to be able to access to certain Module. This defense level will avoid the situation we discussed in the Broken Access Control session. The last check is the privilege level check. The user should not be able to perform administrative actions unless he has full privilege level.

### 3. Session Management

As we mentioned in implementation-level vulnerability Open MRS does not handle session management well enough which does not protect attacker from resuming an abandoned user session. Open MRS is required to log out automatically if the session is inactive for a longer time, and it should be logged out automatically if user closes browser without logging out. However, it does not handle any of them. To mitigate it, You can give hint to the user by popping up a window saying “you may log out before closing the tab” or “you need to quit the browser to logout”. In this case, the user will be inform with a warning message. Also, you can implement code to count inactive session time. If counted time exceed than given limit then system should invalidate the session.

### 4. Input Validation

OpenMRS should have input validation to prevent improper data from entering the database. For example, the name should only be letters. The special characters like this should not be allowed to be a valid name.



5. Dependencies and third-party APIs Check

As we mentioned in phase 1, Open MRS uses many dependencies and third-party APIs with a bunch of serious vulnerabilities. Developers should identify them early enough to reduce unplanned work in the future. The developer should have figured out the security objective of their product and the seriousness of data and its confidentiality before using them. Also, the developer should not always rely on vulnerabilities only mention on the internet. The developer may find new vulnerabilities with time; therefore, no matter what, the developer should use available tools to check dependency. For example, Dependency check is an open source command line tool from OWASP. It supports like Java and Ruby. Also, there is a use of javascript a lot in Open MRS. RetireJS is an open-source JavaScript-specific dependency checker. Snyk is a commercial service that focuses on JavaScript npm dependencies.

In conclusion, we learned various real-world problems of software security. So far, the definition of a successful project was running code with efficiency in terms of time and space; however, we acquired the new definition of a successful project. This project helped us learn about various OWAPS vulnerabilities and mitigation strategies. We have learned the importance of security risk management, security testing, and secure coding techniques. Overall, Open MRS does not have plenty of obvious vulnerabilities; however, it requires many changes as we discussed in the report. Open MRS was an engaging software system since it is designed according to industry standards.