Software Security Project Part 2:
Password, Abuse/misuse cases, Attack trees, Vulnerability History

Name	Unity ID
Abhash Jain	ajain28
Arjun Sharma	asharm33
Sachin Kumar	skumar26
Aishwarya Sundararajan	asundar2

Solution 1:

OpenMRS password policy is as follows:

Maximum password length:

There is no upper limit specified in the OpenMRS code or properties for setting the passwords. Password is stored as hash of 128 characters in database, however in our tests application allowed saving passwords greater than 128 characters.

Minimum password length:

As per the defaults, minimum password length is 8 characters and it can be modified by changing the value of system property **“security.passwordMinimumLength“**

Allowed characters:

Uppercase characters, lowercase characters, atleast one digit, atleast one non-digit

Number of allowed character categories:

uppercase letter - atleast 1
lowercase letter - atleast 1
numbers - atleast 1
Symbols - any

Password age:

There are no properties in the OpenMRS and also no checks in the code checking the password age.

Password reuse policy:

There are no properties in the OpenMRS and also no checks in the code checking the password reuse. We have tried to Manually set one of the old passwords and OpenMRS has allowed us to setup the same old password.

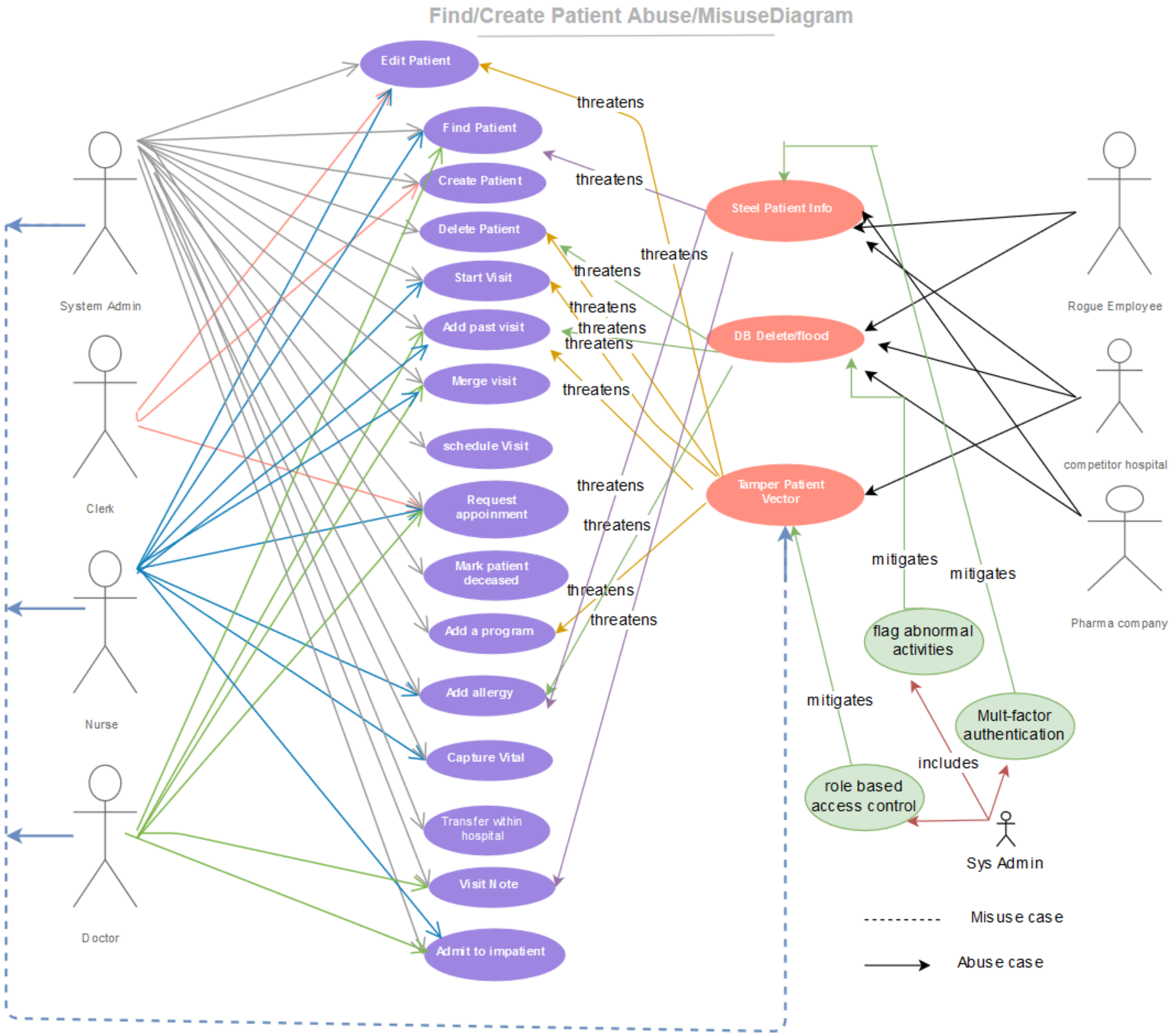
Account lock out:

As per OpenMRS documentation, default number of incorrect logins before lockout is 7, however it can be changed by changing the value of property **“security.allowedFailedLoginsBeforeLockout”.** After max number of incorrect logins, users are locked out for 5 minutes. Ip address are blocked after 10 incorrect username/password attempts.

Reference: **https://wiki.openmrs.org/display/docs/Administering+Users**

Solution 2:

Abuse/ Misuse Case:



1. **Abuse Case : SPI Description:**

- **Name :** Steal Patient Information
- **Summary :** Patient details are extremely sensitive, and someone can do a lot of harm by stealing the patient details. Possible consequences will be data sold to marketers, who then use it to spam the patients.
- **Author :** Sachin Kumar
- **Date :** 10/1/2018
- **Basic Path:**

BP0 : If a username is known, attacker like competitor company or pharma company can try to guess the password with brute-force attacks. Once password is found, attacker can login into the system and can steal patient details.

BP1 : A rogue employee, which can be system admin or clerk or patient or doctor, can login into the system with their credentials and steal the patient details to use it in malicious way.

- **Alternate Path:**
  AP1 : An attacker can obtain password through phishing or social engineering., other than hacking the password

  AP2 : An attacker after doing identity theft of a patient, can successfully trick nurse or doctor to reveal the patient details.

- **Capture Points:**
  CP1 : Patients are authenticated with multi-factors authentication like sending an One time password to email and mobile phone in addition to patient id check before revealing the patient details on request by a patient. (AP2)

  CP2 : Account lockouts after number of failed login attempts, followed by a cooling off period before allowing relogin, prevent attackers usage of brute-force methods to guess password.

- **Extension Points:** None

- **Preconditions:**
  PC1 : System has users mapped to roles like admin,doctor,patient,clerk and then those users are then mapped to some application and organizational privileges.

  PC2 : System allows login of different roles like admin, doctor, patient, clerk into the OpenMRS system.

- **Assumptions:**
  AS1 : User being manipulated by social engineering to reveal the OpenMRS password.

  AS2 : Users like doctor, clerk, nurse and admin has access to certain account privileges with certain permissions to view or edit patient details.

- **Worst Case Threat (Postcondition):**
  WS1 : All of the patient details of OpenMRS system has been compromised

- **Capture Guarantee(Postcondition):**
  CG1 : Patient details data is secure and not hackable by the attacker.

- **Related Business Rules:**
  BR1 : Admin, nurse and clerk have access to view the patient data.

- **Potential Misuser Profile:**
  Someone with the knowledge of hacking and motivation or incentive to attack and misuse the stolen patient details.

- **Stakeholders and Threats:**
  SH1 Doctor/Clerk/Nurse : These roles are under threat as their accounts has been compromised to hack into the openmrs system.

  SH2 Patient : Since its patients details that has been hacked, they are the stakeholders with most impact.

  SH3 Hospital : Reputation of Hospital among its patients will bear a large negative impact, impairing the trust among the patients who have entrusted their personal details with them.

- **Scope:** Find/Create Patient module

- **Abstraction Level:** Attacker goal

- **Precision Level:** Focused

2. **Abuse Case : DBDF Description:**

- **Name :**Database Delete/Flood

- **Summary :** Since different users of OpenMRS has access to enter patient details into the system, which is then stored into database, it is highly likely that someone with malicious intent making use of UI interface ,can delete all the records stored in database or flood database with fake patient details

- **Author :** Arjun Sharma

- **Date :** 10/2/2018

- **Basic Path:**
  BP0 : For a username belonging to the roles of either of admin,doctor and nurse is known with the access to enter or edit the patient details, then attacker can try to guess the password with brute-force attacks. Once password is found,attacker can delete all the patient details or enter the fake patient details.

- **Alternate Path:**
  AP1 : An attacker can obtain password through phishing or social engineering., other than hacking the password

  AP2 : An attacker after doing identity theft of a patient, can successfully trick nurse or doctor to reveal the patient details.

- **Capture Points:**

CP1 : Patients are authenticated with multi-factors authentication like sending an One time password to email and mobile phone in addition to patient id check before revealing the patient details on request by a patient. (AP2)

CP2 : Account lockouts after number of failed login attempts, followed by a cooling off period before allowing relogin, prevent attackers usage of brute-force methods to guess password.

CP3 : Automated monitoring of abnormal activities like larger number of changes to patient details over shorter time period, larger number of edits done by a user for the patients can be be used as precursor to automated flagging or lockdown of that user account.

- **Extension Points:** None

- **Preconditions:**
  PC1 : System has users mapped to roles like admin,doctor,patient,clerk and then those users are then mapped to some application and organisational privileges.

  PC2 : System allows login of different roles like admin,doctor,patient,clerk into the OpenMRS system.

- **Assumptions:**
  AS1 : User being manipulated by social engineering to reveal the openmrs password.

  AS2 : Users like doctor,clerk,nurse and admin has access to certain account privileges with certain permissions to enter or edit patient details.

- **Worst Case Threat (Postcondition):**
  WS1 : All of the patient details of OpenMRS system has been wiped out from the system.

  WS2 : Fake records has flooded the database storage and subsequent exhaustion of storage ,resulting in application availability for real users.

- **Capture Guarantee(Postcondition):**
  CG1 : Patient details data is secure to be deleted by the attacker.

  CG2 : Application is secure to the crash because of the flooding of database possible by fake data insertion by attacker.

- **Related Business Rules:**
  BR1 : Admin and doctor have access to tamper patient data.

- **Potential Misuser Profile:**
  Someone with the motivation or incentive to tarnish the reputation of the functioning medical hospital.

- **Stakeholders and Threats:**
  SH1 Doctor/Clerk/Nurse : These roles are under threat as their accounts has been compromised to the competitive employees.

SH2 Patient : Since its patients details that has been hacked, they are the stakeholders with most impact.

SH3 Hospital : Reputation of the hospital among its patients will bear a large negative impact, impairing the trust among the patients who have entrusted their personal details with them.

- **Scope:** Find/Create Patient module
- **Abstraction Level:** Attacker goal
- **Precision Level:** Focused

3. **Abuse Case : TPV Description:**

- **Name :**Tamper patient vector
- **Summary :** Competitive employees have a motive to tamper with existing patient records in order to disrepute the competing hospital. If patient records are tampered and if the diagnosis recorded for patients are changed, they could be treated for the wrong disease. The incentive is for the patients to leave the current hospital with tampered records and seek medical attention at competitive hospitals.
- **Author :** Aishwarya Sundararajan
- **Date :** 10/3/2018
- **Basic Path:**
  BP0 : BP0 : For a username belonging to the roles of either of admin,doctor and nurse is known with the access to enter or edit the patient details, then attacker can try to guess the password with brute-force attacks. Once password is found,attacker can tamper with the patient details.
- **Alternate Path:**
  AP1 : A competitive employee can threaten Admins or doctors to obtain their credentials

  AP2 : After obtaining the credentials, the employee can enter the system and change patient details and diagnosis.
- **Capture Points:**
  CP1 : Admin, doctors, and nurses are authenticated with multi-factors authentication like sending a One time password to email and mobile phone in addition to verifying their id before revealing any details. If any one of these authentications fail, fail-safe default is the way to go.

  CP2 : Automated monitoring of abnormal activities like larger number of changes to patient details over shorter time period, larger number of edits done by a user for the patients can be be used as precursor to automated flagging or lockdown of that user account.

- **Extension Points:** None

- **Preconditions:**
  PC1 : System has users mapped to roles like admin,doctor,patient,clerk and then those users are then mapped to some application and organisational privileges.

  PC2 : System allows login of different roles like admin,doctor,patient,clerk into the OpenMRS system.

- **Assumptions:**
  AS1 : Admin, Nurse, or Doctor could be bribed or threatened to provide their access credentials.

  AS2 : Users like doctor and admin has access to certain account privileges with certain permissions to edit patient details.

- **Worst Case Threat (Postcondition):**
  WS1 : All of the patient details of OpenMRS system have been tampered. None of the records are accurate. The entire medical system is down.

  WS2 : Loss of trust from patients' perspective. Employees could lose their jobs for good.

- **Capture Guarantee(Postcondition):**
  CG1 : Patient details data is secure to be tampered by the competitive employee..

- **Worst Case Threat (Postcondition):**
  WS1 : Many patient records of OpenMRS system have been tampered.

  WS2 : Loss of trust from patients' perspective. Employees could lose their jobs for good.

- **Capture Guarantee(Postcondition):**
  CG1 : A confirmation alert box just before changing the patients' records may prevent the admin from changing the wrong detail
  CG2: The admin cannot submit wrong information

- **Related Business Rules:**
  BR1 : Admin,nurse and clerk have access to enter/delete patient data.

- **Potential Misuser Profile:**
  Someone with the knowledge of hacking and motivation or incentive to crash the application by flooding the database or deleting the patient details.

- **Stakeholders and Threats:**
  SH1 Doctor/Clerk/Nurse : These roles are under threat as their accounts has been compromised to hack into the openmrs system.

  SH2 Patient : Since its patients details that has been hacked, they are the stakeholders with most impact.

SH3 Hospital
: Reputation of Hospital among its patients will bear a large negative impact, impairing the trust among the patients who have entrusted their personal details with them.

- **Scope:** Delete patient/ Start visit/Add program/Add past visit modules
- **Abstraction Level:** Competitive employee goal
- **Precision Level:** Focused

4. Misuse Case : TPV-Misuse Description:

- **Name :**Tamper patient vector
- **Summary :** System administrator could accidentally tamper patient records or add wrong programs which could affect patient details and diagnosis. This could affect the drugs being administered to the patient and in the worst case cause the patient's condition to worsen.
- **Author :** Abhash Jain
- **Date :** 10/3/2018
- **Basic Path:**
  BP0 : For a username belonging to the roles of either of admin,doctor and nurse is known with the access to enter or edit the patient details, then attacker can try to guess the password with brute-force attacks. Once password is found,attacker can tamper with the patient details.
- **Extension Points:** None
- **Worst Case Threat (Postcondition):**
  WS1 : All of the patient details of OpenMRS system have been tampered. None of the records are accurate. The entire medical system is down.

  WS2 : Loss of trust from patients' perspective. Employees could lose their jobs for good.
- **Capture Guarantee(Postcondition):**
  CG1 : Patient details data is secure to be tampered by the competitive employee.

- **Related Business Rules:**
  BR1 : Admin,nurse and clerk have access to enter/delete patient data.
- **Potential Misuser Profile:**
  Admin who is negligent towards his duties and lacks concentration
- **Stakeholders and Threats:**

SH1 Patient : Since its patients details that has been hacked, they are the stakeholders with most impact.
SH2 Admin: The admin could lose his/her job.
SH3 Hospital : Reputation of Hospital among its patients will bear a large negative impact, impairing the trust among the patients who have entrusted their personal details with them.

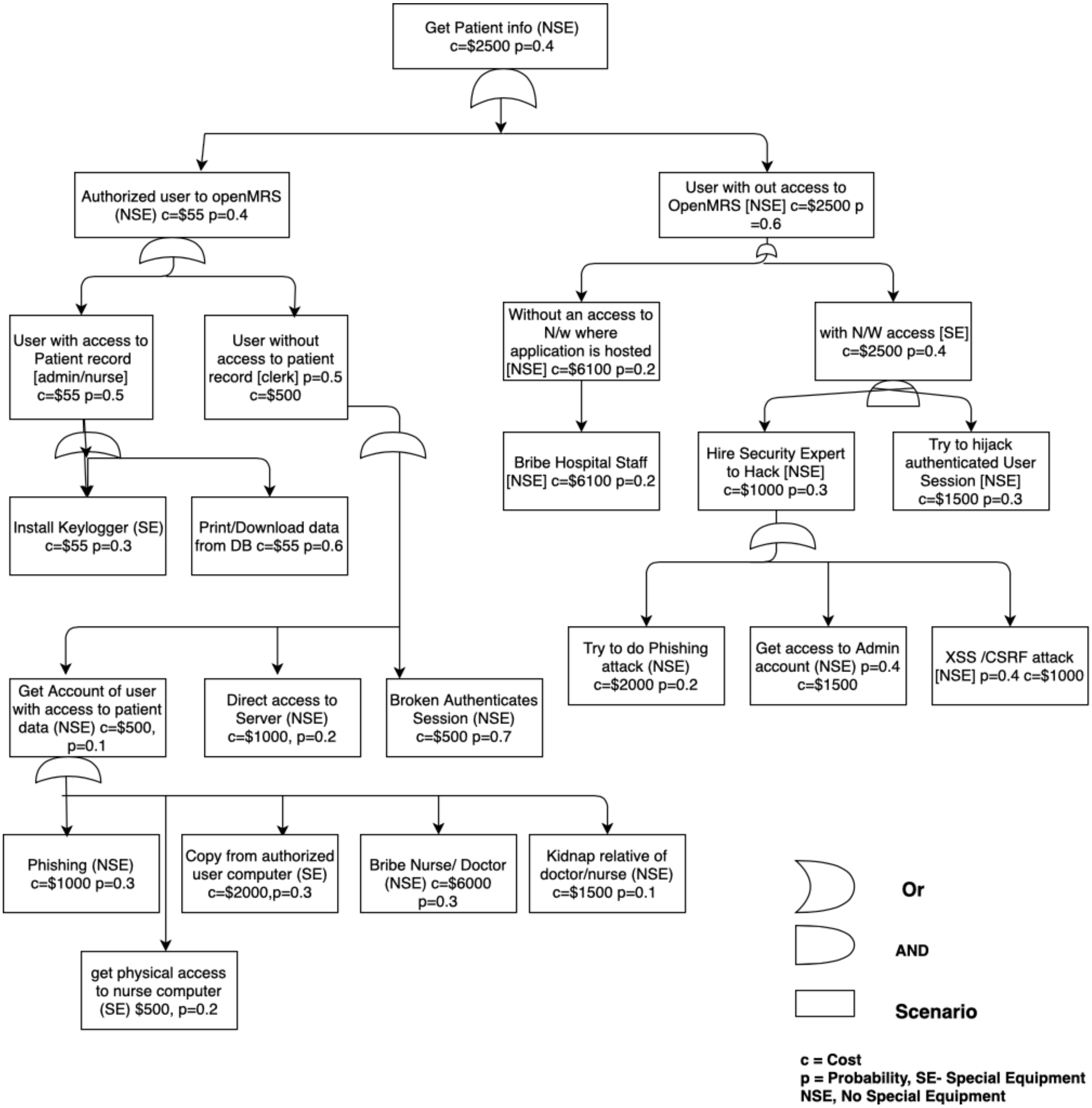
- **Scope:** Delete patient/ Start visit/Add program/Add past visit modules

- **Abstraction Level:** Misuser goal

- **Precision Level:** Focused

Solution 3:

Attack tree to steal the patient record data from Find/create patient module:



Get Patient info (NSE) c=$2500 p=0.4
Authorized user to openMRS (NSE) c=$55 p=0.4
User with out access to OpenMRS [NSE] c=$2500 p =0.6
User with access to Patient record [admin/nurse] c=$55 p=0.5
User without access to patient record [clerk] p=0.5 c=$500
Without an access to N/w where application is hosted [NSE] c=$6100 p=0.2
with N/W access [SE] c=$2500 p=0.4
Bribe Hospital Staff [NSE] c=$6100 p=0.2
Hire Security Expert to Hack [NSE] c=$1000 p=0.3
Try to hijack authenticated User Session [NSE] c=$1500 p=0.3
Install Keylogger (SE) c=$55 p=0.3
Print/Download data from DB c=$55 p=0.6
Try to do Phishing attack (NSE) c=$2000 p=0.2
Get access to Admin account (NSE) p=0.4 c=$1500
XSS /CSRF attack [NSE] p=0.4 c=$1000
Get Account of user with access to patient data (NSE) c=$500, p=0.1
Direct access to Server (NSE) c=$1000, p=0.2
Broken Authenticates Session (NSE) c=$500 p=0.7
Phishing (NSE) c=$1000 p=0.3
Copy from authorized user computer (SE) c=$2000,p=0.3
Bribe Nurse/ Doctor (NSE) c=$6000 p=0.3
Kidnap relative of doctor/nurse (NSE) c=$1500 p=0.1
get physical access to nurse computer (SE) $500, p=0.2
Or
AND
Scenario
c = Cost
p = Probability, SE- Special Equipment
NSE, No Special Equipment

Description and Justification for Attack Tree:

The OpenMRS System can be used to attack to get the personal and Health related information from the system. These information are very critical type of information. Leaking this type of information violates Health Insurance Portability and Accountability Act (HIPAA) in United States and may violated GDPR in EU (European Union). So, protecting this sensitive information become very crucial. From the OpenMRS System we have come across the above mentioned attack tree where main aim of the tree is to steal the patient information.

These attack can be initiated either by an insider to benefit their personal motives or take revenge against their organisation(Hospital). In above tree, attack can be carried out by someone who has authorized access to open system. This include user like Nurse/doctor which have direct access to patient record.

Second, There are users which don't have access to OpenMRS system. But, they still might need patient information. This involves Competitive hospital, Pharma companies to harm the reputation the information. They might try different method to get access the data. To carry out this they may hire some security expert to get the patient information.

Each patient record cost $50 in black market and if rogue user successfully able to steal 1000 records than it sums up to $50000. Which is worth trying this attack scenario for financial gain.

Attack Scenario from a OpenMRS Authorized User:

1. The attacker in this case may be motivated to steal information for financial reasons or take revenge from the organization may be due to less hike. These patient record information can be sell in open market. These patient information can be used to do data analytics and sell their medicine records.
2. User like nurse and doctor who has access to this information may copy the personal information to a drive or print it and sell it. Or they can use keylogger in some other peers computers to put blame on someone else.
3. User like clerk who actually don't have access to the patient record can also try to use them for their personal reasons. They can get the details of other user by taking other user credential try to copy them in some media. They can try to do broken authentication ,but it is going to little difficult from someone who don't know about technology. They can also either try to kidnap or bribe their peers.

Attack Scenario from a OpenMRS UnAuthorized User:

1. In this case user will be motivated to gain financial benefit. It can be a competitive hospital which is trying to harm the reputation of the hospital or make profit to send offer to current active patient. Or it can be a new Pharma company which is trying to sell their product in new market. To study the disease pattern in the area they can try to steal the patient information.
2. As they don't have physical access to the network, so they might try to contact someone working in the hospital and try to bribe them in favor of getting patient personal health record.
3. As this become very difficult to gain access of the Database, they will hire a security expert which is very in depth knowledge about taking these kind of task. Expert somehow try to target a weak point in system by hijacking the session of any naive System user. Once, he get the access to the system then he may try to elevate his privileges by doing stored XSS or CSRF and broken authentication and hijacking the admin session. He can also try to do Phishing attack to gain the access the account of nurse and doctors.

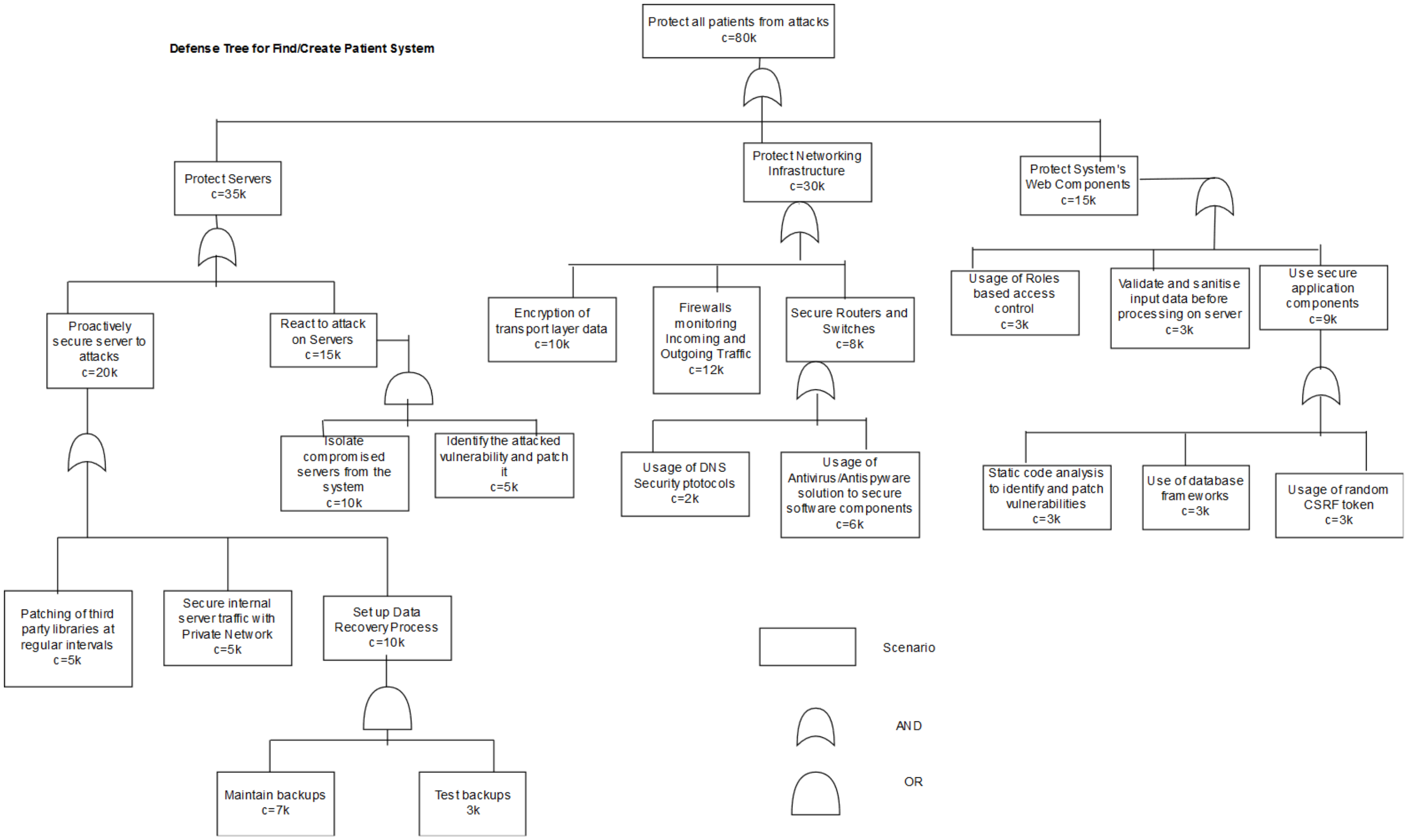
Special Equipment required to attack on the System:

1. **Brute-force attack:** Aircrack-ng, Rainbow crack etc.
2. **Phishing attack:** LUCY, MSI Simple Phis, Gophish etc.
3. **Keylogger:** Hardware or Software based keylogger
4. **Penetration test tools:** Burp suite etc.

Citation for attack tree:

1. Kidnapping cost: https://www.therichest.com/expensive-lifestyle/it-only-costs-1-500-to-get-yourself-kidnapped/
2. Keylogger price: https://www.amazon.com/Keyllama-4MB-Value-Keylogger-black/dp/B004ZGXU48
3. Nurse can be bribe with 1 month salary: https://nursesalaryguide.net/registered-nurse-rn-salary/
4. Attack Cost: https://www.vadesecure.com/en/spear-phishing-cost/
5. https://resources.infosecinstitute.com/popular-tools-for-brute-force-attacks/#gref
6. https://www.skyhighnetworks.com/cloud-security-blog/top-phishing-test-tools-and-simulators/
7. Patient record cost in open market: https://veriphyr.com/patient-data-worth-50-each-on-black-market/

Defense tree to protect the patient record data from Find/create patient module:


Defense Tree for Find/Create Patient System
Protect all patients from attacks c=80k
Protect Servers c=35k
Protect Networking Infrastructure c=30k
Protect System's Web Components c=15k
Proactively secure server to attacks c=20k
React to attack on Servers c=15k
Isolate compromised servers from the system c=10k
Identify the attacked vulnerability and patch it c=5k
Patching of third party libraries at regular intervals c=5k
Secure internal server traffic with Private Network c=5k
Set up Data Recovery Process c=10k
Maintain backups c=7k
Test backups 3k
Encryption of transport layer data c=10k
Firewalls monitoring Incoming and Outgoing Traffic c=12k
Secure Routers and Switches c=8k
Usage of DNS Security ptotocols c=2k
Usage of Antivirus/Antispyware solution to secure software components c=6k
Usage of Roles based access control c=3k
Validate and sanitise input data before processing on server c=3k
Use secure application components c=9k
Static code analysis to identify and patch vulnerabilities c=3k
Use of database frameworks c=3k
Usage of random CSRF token c=3k
Scenario
AND
OR

Description and Logical Explanation:

OpenMRS's infrastructure,servers and web applications components should be secured in order to protect patient details from attacks.

System web application components can be secured by implementing roles based access control, which can limit the privileges to the account based on roles assigned to userid's. Also since unsanitised data can make application prone to XSS or Cross-site scripting attacks, so all the data should be sanitised before processing on server side.Futher usage of secure application components can secure the web application components. In the same context,database frameworks like Hibernate can secjure the application from SQL injection attacks. Static code analyser can help find all the application vulnerabilities at implementation phase and prevent propagation of security bugs and flaws. Also random CSRF token can be used to secure application against CSRF(Cross-site Request Forgery) attacks.

Networking infrastructure also serves as one of the weak link for attacker and need to be secured.In the same direction certain measures can be adopted. Firstly,encryption of transport layer data can be done by SSL/TLS protocol to ensure data in transit is secure to eavesdropping. Secondly,firewalls should be configured to monitor the incoming and outgoiung network packets for suspicious activity.Thirdly,routers and switches should be secured, with use of antivirus/antispyware applications to secure it against the network intrusions of viruses and spyware,also dns security protocols like DNSSEC(DNS Security Extensions) should be used to protect dns server attacks by digitally signing data to help ensure its validity.

Servers being one of the cornerstone of OpenMRS system should be secured against the attacks by adoptiong different measures based on particular scenarios.In the first scenario of Proactive securing of server attacks, different measures can be adopted.Firstly all the third party libraries should be updated at regular intervals to ensure that security vulberabilities are addressed in the system with the latest patches released by third party provider.Secoondly, internal server traffic can be secured with private network to make it immune against the attacks by remote attackers over the network.Lastly, a data recovery proces can be setup to ensure the data is not lost after a attack and it can be rolled back to the backup copy of data , in case data is lost during attack.For same, it need to be ensured that regular data backups are maintained and also tested during the same process if data backup generated is integral and true replica of the patient data it was configured for.

Special Equipment:

1. Antivirus/Antispyware softwares like BitDefender,Norton.
2. VPC(Virtual Private Cloud) cluster for private cloud
3. Enterprise data backup solutions like Veritas
4. Static code analyser tools like Sonargraph
5. Database framework like Hibernate

Citations for defense tree:

1. DNS Security : https://www.cloudflare.com/learning/dns/dns-security/
2. VPC Cloud : https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html
3. Data backup and recovery : http://www.enterprisestorageforum.com/storage-networking/enterprise-backup-and-recovery-management.html
4. CSRF attacks security : https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
5. Static code analysis : https://www.owasp.org/index.php/Static_Code_Analysis
6. Transport layer Security : https://hpbn.co/transport-layer-security-tls/

Solution 4:

Vulnerability History:

1) Reporting Compatibility Add On Vulnerability: CVE-2017-12796

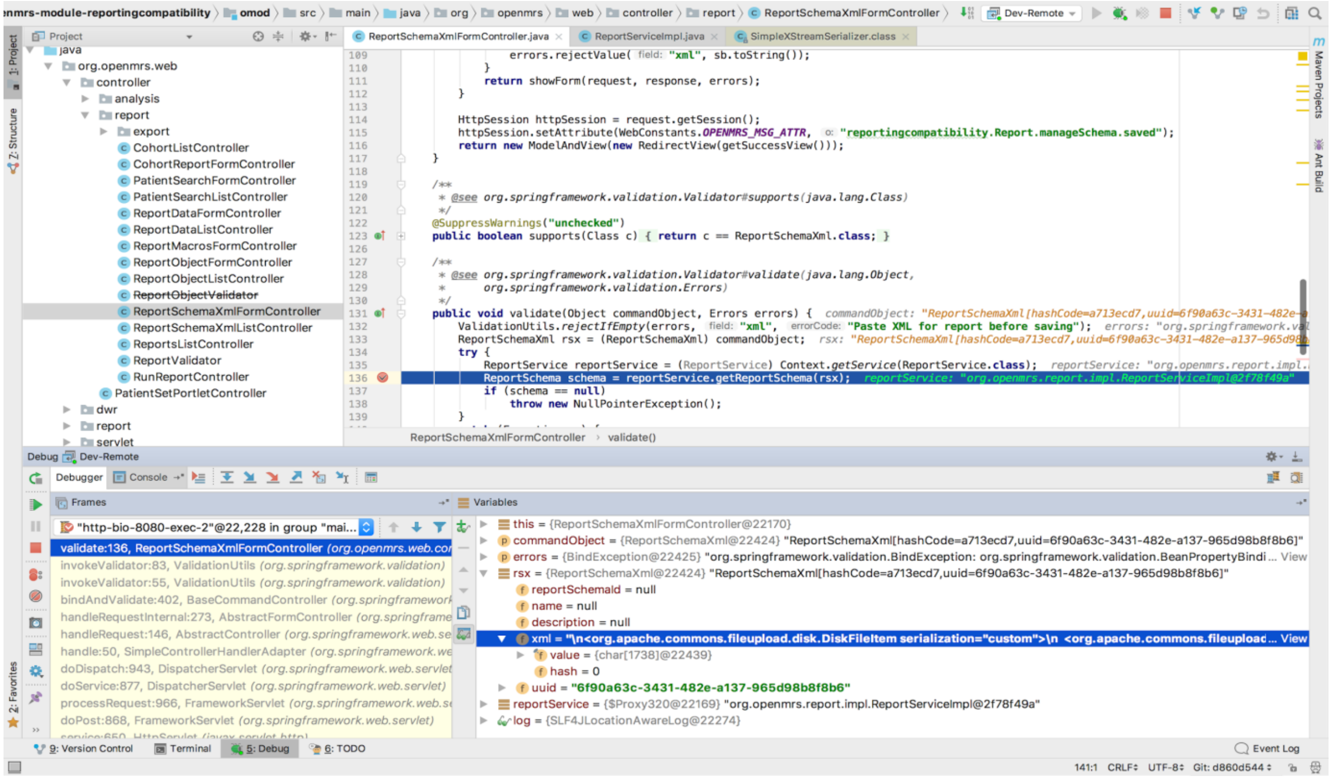
a) The Reporting Compatibility Add On before 2.0.4 for OpenMRS, as distributed in OpenMRS Reference Application before 2.6.1, doesn't authenticate users when deserializing XML input into ReportSchema objects.

b) 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.

c) CVSS Score: 10.0

d) Exploitation of this vulnerability is possible through a single HTTP POST request to the page at http://152.46.17.224:8082/openmrs-standalone/admin/reports/reportSchemaXml.form

e) Accessing this page through a browser without authenticating first will redirect the user to the login page (so far so good). Under the hood, however, the application is actually executing server-side code before the HTTP redirect response is generated (not so good). Through a Java debugger, with a few strategically placed breakpoints, it becomes apparent that a validation function is being called prior to any auth checks in the reportSchemaXml form controller:


enmrs-module-reportingcompatibility › omod › src › main › java › org › openmrs › web › controller › report › ReportSchemaXmlFormController
Dev-Remote
Project
ReportSchemaXmlFormController.java
ReportServiceImpl.java
SimpleXStreamSerializer.class
1: Project
7: Structure
2: Favorites
Maven Projects
Ant Build
java
org.openmrs.web
controller
analysis
report
export
CohortListController
CohortReportFormController
PatientSearchFormController
PatientSearchListController
ReportDataFormController
ReportDataListController
ReportMacrosFormController
ReportObjectFormController
ReportObjectListController
~~ReportObjectValidator~~
ReportSchemaXmlFormController
ReportSchemaXmlListController
ReportsListController
ReportValidator
RunReportController
PatientSetPortletController
dwr
report
servlet
ReportSchemaXmlFormController › validate()
Debug Dev-Remote
Debugger
Console
Frames
Variables
9: Version Control
Terminal
5: Debug
6: TODO
Event Log
141:1 CRLF UTF-8 Git: d860d544

```
                errors.rejectValue( field: "xml", sb.toString());
            }
            return showForm(request, response, errors);
        }

        HttpSession httpSession = request.getSession();
        httpSession.setAttribute(WebConstants.OPENMRS_MSG_ATTR,  o: "reportingcompatibility.Report.manageSchema.saved");
        return new ModelAndView(new RedirectView(getSuccessView()));
    }

    /**
     * @see org.springframework.validation.Validator#supports(java.lang.Class)
     */
    @SuppressWarnings("unchecked")
    public boolean supports(Class c) { return c == ReportSchemaXml.class; }

    /**
     * @see org.springframework.validation.Validator#validate(java.lang.Object,
     *      org.springframework.validation.Errors)
     */
    public void validate(Object commandObject, Errors errors) {   commandObject: "ReportSchemaXml[hashCode=a713ecd7,uuid=6f90a63c-3431-482e-a
        ValidationUtils.rejectIfEmpty(errors,  field: "xml",  errorCode: "Paste XML for report before saving");   errors: "org.springframework.val
        ReportSchemaXml rsx = (ReportSchemaXml) commandObject;   rsx: "ReportSchemaXml[hashCode=a713ecd7,uuid=6f90a63c-3431-482e-a137-965d98b
        try {
            ReportService reportService = (ReportService) Context.getService(ReportService.class);   reportService: "org.openmrs.report.impl.
            ReportSchema schema = reportService.getReportSchema(rsx);   reportService: "org.openmrs.report.impl.ReportServiceImpl@2f78f49a"
            if (schema == null)
                throw new NullPointerException();
        }
```

```
"http-bio-8080-exec-2"@22,228 in group "mai...
validate:136, ReportSchemaXmlFormController (org.openmrs.web.con
invokeValidator:83, ValidationUtils (org.springframework.validation)
invokeValidator:55, ValidationUtils (org.springframework.validation)
bindAndValidate:402, BaseCommandController (org.springframework
handleRequestInternal:273, AbstractFormController (org.springframe
handleRequest:146, AbstractController (org.springframework.web.se
handle:50, SimpleControllerHandlerAdapter (org.springframework.we
doDispatch:943, DispatcherServlet (org.springframework.web.servlet
doService:877, DispatcherServlet (org.springframework.web.servlet)
processRequest:966, FrameworkServlet (org.springframework.web.se
doPost:868, FrameworkServlet (org.springframework.web.servlet)
service:650, HttpServlet (javax.servlet.http)
```

```
▶ this = {ReportSchemaXmlFormController@22170}
▶ commandObject = {ReportSchemaXml@22424} "ReportSchemaXml[hashCode=a713ecd7,uuid=6f90a63c-3431-482e-a137-965d98b8f8b6]"
▶ errors = {BindException@22425} "org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindi... View
▼ rsx = {ReportSchemaXml@22424} "ReportSchemaXml[hashCode=a713ecd7,uuid=6f90a63c-3431-482e-a137-965d98b8f8b6]"
      reportSchemaId = null
      name = null
      description = null
   ▼ xml = "\n<org.apache.commons.fileupload.disk.DiskFileItem serialization="custom">\n  <org.apache.commons.fileupload... View
      ▶ value = {char[1738]@22439}
        hash = 0
   ▶ uuid = "6f90a63c-3431-482e-a137-965d98b8f8b6"
▶ reportService = {$Proxy320@22169} "org.openmrs.report.impl.ReportServiceImpl@2f78f49a"
▶ log = {SLF4JLocationAwareLog@22274}
```

f) The end result is still a HTTP 302 to the login page. The real problem here is revealed by stepping into the call to **reportService.getReportSchema(rsx),** which was identified by **Isaac Sears** and his team.'

g) This is a common recurring vulnerability CWE-502: Deserialization of Untrusted Data.

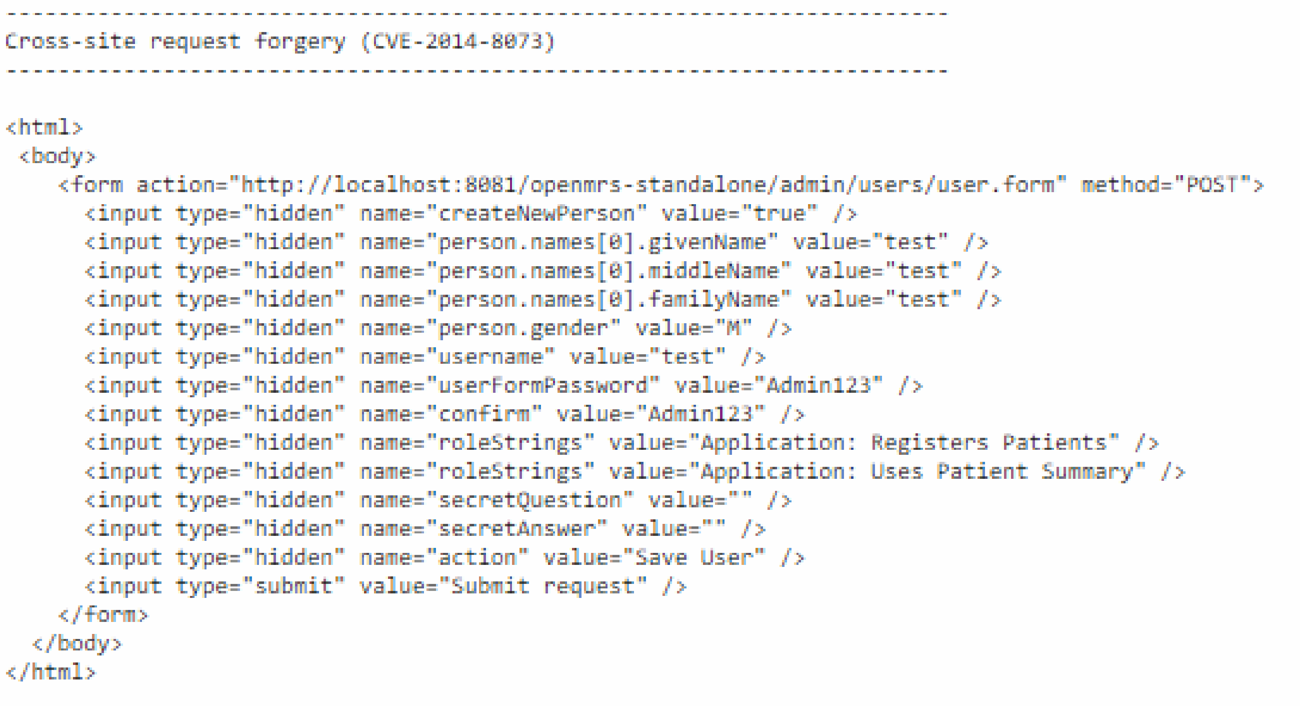
h) Mitigation Steps: Anyone running the html form entry or reporting compatibility module (included in the Reference Application) should immediately upgrade to the latest released versions of the modules, which are available on **https://addons.openmrs.org/**.

Reference: https://isears.github.io/jekyll/update/2017/10/21/openmrs-rce.html

2) Cross-site request forgery (CSRF): CVE-2014-8073

a) 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.

b) Exploring this vulnerability may allow an attacker to bypass certain security restrictions, obtain sensitive information, execute arbitrary script code in the browser of an unsuspecting user, steal cookie-based authentication credentials, and perform actions in the vulnerable application in the context of the victim.



```
-----------------------------------------------------------------------
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>
```

c) CVSS Score: 6.8

d) This is a common recurring vulnerability CWE-352: Cross-Site Request Forgery (CSRF).

e) Mitigation Steps:

- Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
- Ensure that the application is free of cross-site scripting issues (CWE-79), because most CSRF defenses can be bypassed using attacker-controlled script.
- Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable
- Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.
- Do not use the GET method for any request that triggers a state change.
- Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Reference: https://exchange.xforce.ibmcloud.com/vulnerabilities/97692
https://www.securityfocus.com/bid/70664
https://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html

3) Administration module: CVE-2014-8072

a) The administration module in OpenMRS 2.1 Standalone Edition allows remote authenticated users to obtain read access via a direct request to /admin.

b) OpenMRS 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.

c) CVSS Score: 4.0

d) This is a common recurring vulnerability CWE-264: Permissions, Privileges, and Access Controls.

e) Mitigation Steps:

- Admins are advised to apply the appropriate updates and allow only trusted users to have network access.
- Users are advised not to visit websites or follow links that have suspicious characteristics or cannot be verified as safe.
- Admins are advised to use an unprivileged account when browsing the Internet and are advised to monitor critical systems.

Reference: http://cwe.mitre.org/data/definitions/264.html
https://exchange.xforce.ibmcloud.com/vulnerabilities/97693

4) Multiple cross-site scripting: CVE-2014-8071

a) These 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; the (5) comment parameter to allergyui/allergy.page; the (6) w10 parameter to htmlformentryui/htmlform/enterHtmlForm/submit.action; the (7) HTTP Referer Header to login.htm; the (8) returnUrl parameter to htmlformentryui/htmlform/enterHtmlFormWithStandardUi.page or (9) coreapps/mergeVisits.page; or the (10) visitId parameter to htmlformentryui/htmlform/enterHtmlFormWithSimpleUi.page.

b) 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.

c) CVSS Score: 4.3

d) This vulnerability is usually introduced during Architecture and Design, Implementation phase.

e) This is a common recurring vulnerability CWE - 79 : Failure to Preserve Web Page Structure.

f) Mitigation Steps:

- Use languages, libraries, or frameworks that make it easier to generate properly encoded output.
- Understand the context in which your data will be used and the encoding that will be expected.
- Use and specify a strong character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page.
- With Struts, you should write all data from form beans with the bean's filter attribute set to true.

Reference: https://www.cvedetails.com/cwe-details/79/cwe.html
https://www.securityfocus.com/bid/70664/info
https://exchange.xforce.ibmcloud.com/vulnerabilities/97690