Password Strength, Abuse/Misuse cases, Attack Trees, Vulnerability History

Project Phase II

Ananthram Eklaspuram Lakshmanasamy - aeklasp
Arun Jaganathan - ajagana
Bhaskar Sinha - bsinha
Nivedita Natarajan – nnatara2

1. Password Strength

Minimum Password Length

The minimum password length that is required while creating accounts on OpenMRS is 8 characters.
We can see that, the application asks us to create passwords that are atleast 8 characters in length. This can also be modified in the “Manage Global Properties” section, by changing the value of the field **security.passwordMinimumLength.**

Maximum Password Length

There is no maximum password length that is specifically mentioned in the OpenMRS application. The password is stored as a hash of 128 characters in size. Eventhough this is done, passwords of length greater than 128 characters can also be stored.
We tried passwords inputs of length 300-400 and the application still allowed passwords of such lengths.

Allowable Characters

The types of characters that are allowed are Uppercase, Lowercase alphabets, numbers and symbols. There are however, some required character categories required which is discussed in the following heading.
Passwords including all of these characters were tried manually and checked.

Number of Allowable Character Categories Required

1. Uppercase Characters - atleast 1
2. Lowercase Characters - atleast 1
3. Number - atleast 1
4. Symbol - Any number

Also, the password cannot match the username.
In “Manage Global Properties”, we can see properties such as security.passwordCannotMatchUsername (Assigned to True), security.passwordMinimumLength (Assigned to 8), security.passwordRequiresDigit (Assigned to True), security.passwordRequiresNonDigit (Assigned to True), security.passwordRequiresUpperAndLowerCase (Assigned to True) are all defined. These properties are editable and can be modified according to choice. Additionally, security.passwordCustomRegex can also be defined.

Password Age and Reuse Policy

There does not seem to be any Password Age or Reuse Policy mentioned in the Global Properties section or the code of the OpenMRS as well. Both the sections were gone through and the password age and reuse policy was not to be found in these modules.

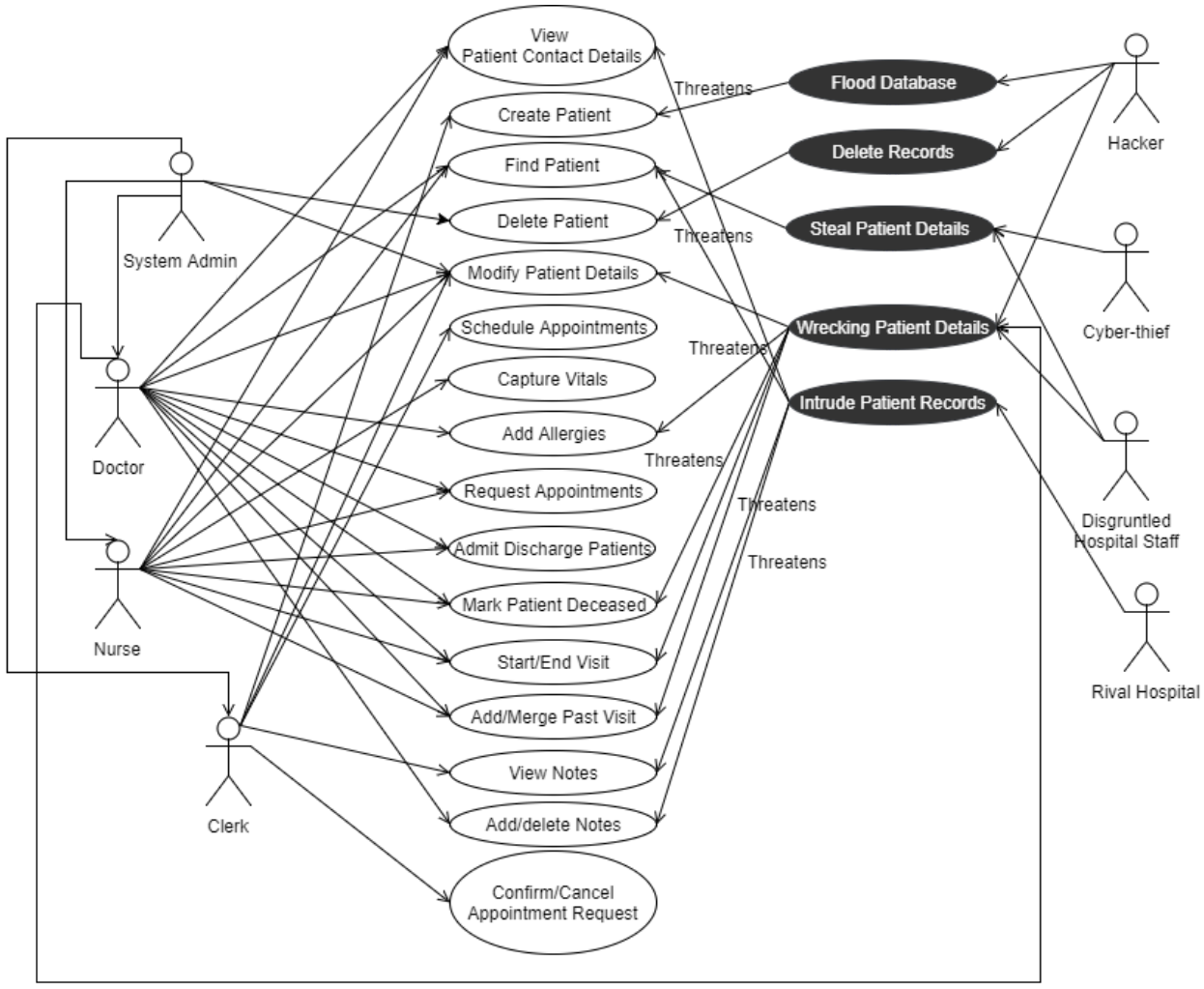
Account Lockout

The documentation of OpenMRS application, specifies that users are locked out for 5 minutes after 7 failed attempts of logging in. The number of failed login attempts can be modified by the global property security.allowedFailedLoginsBeforeLockout. IP addresses are blocked after 10 failed username/password attempts.

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

2. Abuse/ Misuse Case Diagram

For this section, we plan to use ‘Find/Create Patient’ Module to build the Abuse/Misuse Cases. The attack and protection trees are also based on this module.



Abuse/Misuse Cases:

1. Abuse Cases: A-SPD-ET Description:

- **Name:** Steal Patient Details - External Threat
- **Summary:** Medical Data is very important that, if it is accessible to wrong hands, the possibilities of its misuse is very high. Any outsider, say a thief or a hacker may steal the patient record to achieve some purpose. In this abuse case, we are discussing about the thief.
- **Author:** Arun Jaganathan

- **Date:** 10/01/2017

- **Basic Path:**

  BP0 Knowing the username having access to 'Find patient records', the thief tries to enter the password by brute-force method. In OpenMRS, Doctors, Nurses and System Administrators have access to find patient records (Step BP0-1) Once the password is found,the hacker/thief can log-in to the system. (Step BP0-2). After logging in, the thief has access to the patient details, and can decide to sell the information for money (Step BP0-3).

- **Alternate Path:**

  AP1 The hacker/thief can obtain password through other means than hacking.The thief can trick the user to hint their password, an example being phishing (changes step BP0-1).

  AP2 The thief can call/email the hospital and impersonate a patient to access his/her records from the nurse/clerk.

  AP3 The thief can enter the hospital building and can access the system that is left logged into OpenMRS.

- **Capture Points:**

  CP1 Session timeout prevents accessing the OpenMRS (AP3)

  CP2 Account lockout prevents thief from using brute force method (BP0-1)

  CP3 Emails from unknown sources are generally filtered out before it reaches the employee (AP2)

  CP4 Calls from unknown sources are first identified using some sensitive information before data is given out (AP2)

- **Extension Points:** None

- **Preconditions:**

  PC1 The system has privileges mapped to the roles and users mapped to these roles.Each role has a set of organizational and application privileges assigned to it.

  PC2 OpenMRS allows the login of above mentioned employees into the system.

- **Assumptions:**

  AS1 The user falls prey to the tricks used by the hacker/thief. (AP1)

AS2 Certain user roles like Doctor, Nurse and Admin have access to view the patient details.

- **Worst Case Threat (Postcondition):**
  WS1 All of Patient's data has been compromised.

- **Capture Guarantee (Postcondition):**
  CG1 The thief/hacker is not able to get hands on the patient data.

- **Related Business Rules:**
  BR1 The Doctor, Nurse, Admin roles all have access to view the patient data.

- **Potential Misuser Profile:**
  Anyone with the knack of hacking/phishing, who is highly motivated to misuse the patient details.

- **Stakeholders and Threats:**
  SH1 Patients: His/her personal details has been breached.
  SH2 Doctors/Nurse: Threats to these roles as it was their user ID that has been exploited.
  SH3 Hospital: Loses its reputation and trust in people.

- **Scope:** Find/Create Patient Module
- **Abstraction Level:** Thief/Hacker goal
- **Precision Level:** Focussed

2. Abuse Case: A-FD Description:

- **Name:** Flood Database
- **Summary:** In OpenMRS, Clerk and Admin have access to Create/ Register Patient Record. It is thus possible for someone with criminal intent to create multiple fake patient IDs which could lead to database overflow. This could be a person who is a part of terrorist organization or being funded by rival hospital.
- **Author:** Ananthram Eklaspuram Lakshmanasamy
- **Date:** 10/02/2017

- **Basic Path:**
  BP0 Knowing the username having access to 'Create patient records', the hacker tries to enter the password by brute-force method. In OpenMRS, Clerk and System Administrators have access to Create patient records (Step BP0-1) Once the password is found,the hacker can log-in to the system. (Step BP0-2). After logging in, the hacker has access to the create fake patient details form, and can decide to enter random data to fill in the database(Step BP0-3).

- **Alternate Path:**
  AP1 The hacker can obtain password through other means than hacking.The thief can trick the user to hint their password, an example being phishing (changes step BP0-1).
  AP2 The Hacker can enter the hospital building and can access the system that is left logged into OpenMRS.

- **Capture Points:**
  CP1 Session timeout prevents accessing the OpenMRS (AP2)
  CP2 Account lockout prevents thief from using brute force method (BP0-1)
  CP3 There could be checks that monitor the data flowing into database, and can lock out the user when trying to create mass amount of patient data (BP0-3).
  CP4 Medical records could be cross verified with other government approved ID (like, Driver License) whenever new patient is created (BP0-3)

- **Extension Points:** None

- **Preconditions:**
  PC1 The system has privileges mapped to the roles and users mapped to these roles.Each role has a set of organizational and application privileges assigned to it.
  PC2 OpenMRS allows the login of above mentioned employees into the system.

- **Assumptions:**
  AS1 The user falls prey to the tricks used by the hacker/thief. (AP1)
  AS2 Certain user roles like Clerk and Admin have access to create the patient details.
  AS3 Creation of large amount of Fake Data of patient crashes the system

- **Worst Case Threat (Postcondition):**
  WS1 The database crashes due to overflow of data.

- **Capture Guarantee (Postcondition):**
  CG1 The hacker is not able to get hands on the patient data.

- **Related Business Rules:**
  BR1 The Doctor, Clerk and Admin roles all have access to create the patient data.

- **Potential Misuser Profile:**
  Anyone with knack of hacking/phishing and with criminal intent.

- **Stakeholders and Threats:**
  SH1 Patients: The medical records of the existing patients has been corrupted.
  SH2 Doctors/Clerk: Threats to these roles as it was their user ID that has been exploited.
  SH3 Hospital: Loses its reputation and health data and thereby, the trust in people.

- **Scope:** Find/Create Patient Module
- **Abstraction Level:**Hacker goal
- **Precision Level:** Focussed

3. Abuse Cases: A-SPD-IT Description:

- **Name:** Steal Patient Details - Internal Threat
- **Summary:** Important details of patients from the OpenMRS database could be compromised by a disgruntled employee and could be leaked elsewhere. This employee could be a doctor, nurse or any staff of the organization. This information if leaked, could cause a huge threat to the patients whose important details are at stake and could be misused.
- **Author:** Nivedita Natarajan
- **Date:** 10/03/2017

- **Basic Path:**

BP0 The disgruntled hospital staff tries to steal confidential information from the OpenMRS database (unauthorized access) by hacking the “Find Patient Records” module through Injection and Cross-Site Scripting attacks(XSS) (Step bp0-1). The staff then finds a particular set of patient records to steal. (Step bp-0-2). These records could now be leaked online and the details could be exposed (Step bp-0-3-1) or the records could be utilized to get personal information of patient (medical history, SSN etc.) (Step bp-0-3-2).

- **Alternate Path:**
  - AP1 The employee who has the required access to records, takes ransom amount from other people(eg. criminals) to expose confidential information about the patient. This step is different from Step bp-0-1 since no hacking is involved to retrieve the required information.
  - AP2 The disgruntled hospital employee accesses “Find Patient Records” page and gets the patient records of the required victim by searching.

- **Capture Points:**
  - CP1 Injection/XSS attacks do not work because the application sanitizes the inputs before executing them. (BP0-1)
  - CP2 For certain user, like nurse, access to patient records are visible only for those patients who are currently ‘visiting’ for treatment.(BP-0-2)
  - CP4 “Find Patient Record” is protected and cannot be accessed directly by anyone in the application, rather only by authorized users. (AP2)

- **Extension Points:** None

- **Preconditions:**
  - PC1 The system has privileges mapped to the roles and users mapped to these roles.Each role has a set of organizational and application privileges assigned to it.
  - PC2 OpenMRS allows the login of above mentioned employees into the system.

- **Assumptions:**
  - AS1 Patient records include confidential information which should not be exposed.
  - AS2 The hospital staff like Doctor, Nurse, Admin have access to view the patient records.

- **Worst Case Threat (Postcondition):**
  WS1 All of Patient's data has been compromised and the important information Is leaked/exposed.

- **Capture Guarantee (Postcondition):**
  CG1 The disgruntled hospital employee does not get access to the Patient Records.

- **Related Business Rules:**
  BR1 The Doctor, Nurse, Admin roles all have access to view the patient data.

- **Potential Misuser Profile:**
  Any unhappy employee or someone with low moral standards, who could be deceived for money.

- **Stakeholders and Threats:**
  SH1 Patients: His/her personal details has been breached.
  SH2 Hospital: Loses its reputation and trust in people.

- **Scope:** Find/Create Patient Module
- **Abstraction Level:** Misuser goal
- **Precision Level:** Focussed

4. Misuse Cases: M-WPD Description:

- **Name:** Wreck Patient Details
- **Summary:** Important details of patients in the OpenMRS database could be modified by the Doctor unintentionally, like adding allergies for a patient even when they are not relevant for a particular person. This information could result in the wrong intake of medicines by the patient and end up being harmful or even fatal for the patient.
- **Author:** Bhaskar Sinha
- **Date:** 10/03/2017

- **Basic Path:**

  BP0 The doctor logs onto the OpenMRS application and finds a particular patient record (Step bp0-1). The doctor then tries to add allergies for a patient based on the hospital visit. (Step bp-0-2). The doctor adds irrelevant allergies to patient unintentionally. (Step bp-0-3). Unnecessary medication is given to the patient which may prove to be harmful or fatal. (Step bp-0-4).

- **Alternate Path:**

  AP1 The doctor might accept ransom from an attacker and record unnecessary problems/allergies for the patient.

  AP2 The doctor opens the wrong patient record unknowingly and records the allergies or problems.

- **Capture Points:**

  CP1 The doctor does not receive access to the patient records in the OpenMRS application/database. (BP0-1)

  CP2 The doctor is not able to modify or add allergies to a patient during patient visit. (BP-0-2)

- **Extension Points:** None

- **Preconditions:**

  PC1 OpenMRS allows the login of the doctor and addition/modification of allergies for a patient record.

- **Assumptions:**

  AS1 Allergies or problems are recorded for a patient by a Doctor upon visit.

  AS2 The corresponding medication for each problem is prescribed to the patient and is consumed by him/her.

- **Worst Case Threat (Postcondition):**

  WS1 The patient is prescribed unwanted medication which may harm the patient or make it fatal.

- **Capture Guarantee (Postcondition):**

  CG1 The doctor is not able to add erroneous allergies for a patient.

  CG2 A confirmation alert that comes up may stop the doctor from adding an erroneous allergy.

- **Related Business Rules:**
  BR1 The Doctor has all access to view and modify patient data and record allergies.

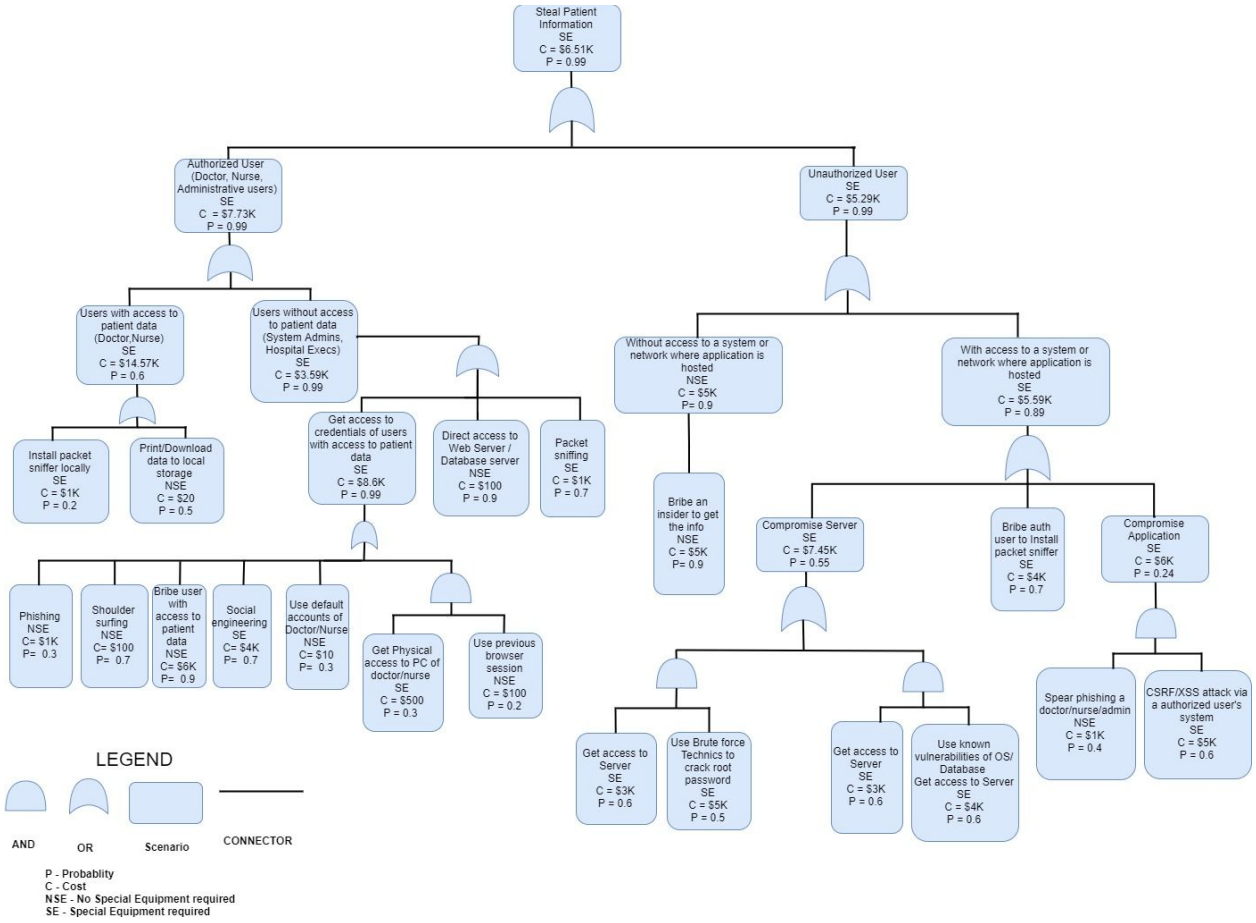
- **Potential Misuser Profile:**
  Doctors who are not very careful about handling the patient data.

- **Stakeholders and Threats:**
  T1 Patients: The patient's life is under threat because of the wrong medication.
  SH1 Doctors: The responsibility of the doctor is being misused because of his/her carelessness.
  SH2 Hospital: Loses its reputation and trust among new people who are looking for a hospital.

- **Scope:** Find Patient Module - Allergy Addition
- **Abstraction Level:** Misuser goal
- **Precision Level:** Focussed

3. Attack Trees and Protection Trees

Attack Tree to steal patient data from Find/Create Patient module



Description and Justification of the attack tree

The OpenMRS system in a Hospital can be attacked to steal patient data by basically two sets of people. One, anyone who is a authorized user with access to the OpenMRS system, this group can be further classified as users with direct access to patient records like doctors/nurses and users who don't have direct access to patient records, these include Admin, Executives etc.

Two, anyone who does not have access to the OpenMRS system, this second set can be further divided into people who have access to network where OpenMRS system is hosted, for example Janitors in hospital etc and another is people who are totally outside the OpenMRS network or system premise, for example outside hackers,

another competing hospital etc.

Attack Scenario from a OpenMRS Authorized User:

- The attacker may be motivated to steal patient medical records for mostly monetary reasons, the doctor, nurse or admin can sell the records in the black market or they can steal it for getting a bribe from any other person.
- If the attacker is doctor or nurse (users with official access to medical records), their easiest and cheapest way to steal data is to just use the find patient module and download the data (to a storage device or print it). Another way is to install a packet sniffer (special equipment) at their PC but that would cost a lot more and would need more technical expertise,so it is less probable.
- If the attacker is admin (user without official access to medical records), their easiest and cheapest way to steal data if they have access to the Database(DB) server used by OpenMRS is then to directly copy the data from the DB server . The next easy and cheap way to steal the data is to get access to PC of Doctors/nurses and use their previous browser session to download the data, but this method is a little difficult to orchestrate. The other easy way is to directly bribe a doctor/nurse to get the medical records for them, but it would cost a lot more (we predict about $5000, a month's salary of nurse). Other possible ways are to use packet sniffers or use phishing or social engineering techniques to obtain doctor/nurse account credentials.

Attack Scenario from a OpenMRS Unauthorized User:

- Again the attacker may be motivated to steal patient medical records for mostly monetary reasons, for example the janitor (someone with access to premises where OpenMRS is hosted) can sell the records in the black market or they can steal it for getting a bribe from any other person, a self operating hacker can steal patient records to sell in black market or on the instruction of a competing hospital.
- If the attacker does not have access to the network where OpenMRS is hosted or to the premise where servers are located, then the only option is to bribe an insider to get the Medical records, this does not need any special equipment from the attacker's side but will cost around $5000 (a month's salary of nurse).
- If the attacker has access to network where OpenMRS is hosted or to the premise where the servers are set up (this also includes bot/worms/CSRF

attacks followed by phishing mails to authorized members (doctors,nurses)), then the possible ways to steal records are Compromise the Servers by brute force/rainbow table attacks or using any known Operating System or Database vulnerabilities, compromise application by setting up a spear phishing attack on a doctor or nurse and then installing a bot or executing a XSS attack to steal data using the doctor or nurse credentials, bribing an insider to get the records or to install a packet sniffer in the network. All the mentioned ways to steal data involve high costs, but the technique of bribing an insider does not require any special equipment while the other two need. (Cain and Abel, RanibowCrack,Wfuzz for brute force attacks and bot tools etc). So bribing an insider is the preferred method.

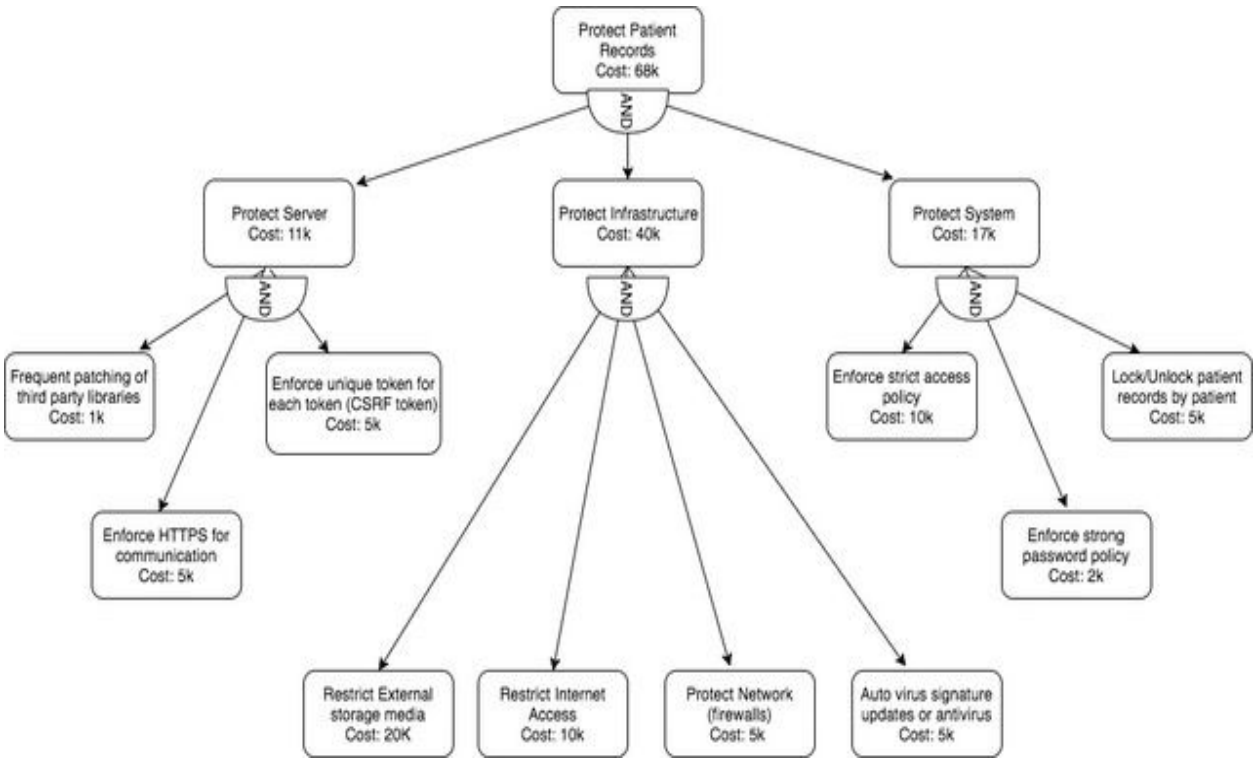
Special Equipments that may be used:

- Brute force attacks: Cain and Abel, RainbowCrack, Aircrack-ng, THC Hydra.
- Phishing: SNAP_R, MSI Simple Phish, SET, Metasploit.
- Packet Sniffers: SharkTap Network Sniffer, PRTG, omnipeek

Evidence/Citations

1. Minimum Cost of bribing a Nurse is assumed as 1 Month salary (US average) of the Nurse(Average salary Details http://nursesalaryguide.net/registered-nurse-rn-salary/).
2. Average monthly salary of a security gaurd in the US http://swz.salary.com/SalaryWizard/Security-Guard-Monthly-Salary-Details.aspx.
3. Cost of making an CSRF/ XSS attack is assumed as ¼ the cost of 2000 medical records at $10/record is $5000.Cost of a medical record is obtained from following link http://www.reuters.com/article/us-cybersecurity-hospitals/your-medical-record-is-worth-more-to-hackers-than-your-credit-card-idUSKCN0HJ21I20140924
4. Cost associated with packet sniffer tools are provided from this wikipedia page https://en.wikipedia.org/wiki/Comparison_of_packet_analyzers
5. Default accounts vulnerability https://technet.microsoft.com/en-us/library/hh825104.aspx
6. Social Engineering and password hacks costs are estimated from these sites http://blog.willis.com/2016/01/social-engineering-is-bigger-than-hacking-but-countermeasures-work/, https://blog.my1login.com/blog/staggering-cost-password-hacks

Protection Tree:



Description and Logical Explanation:

OpenMRSs' IT infrastructure and servers need to be protected in order to protect patient records. To prevent unauthorized access to the system strict access policies should be enforced to prevent access of patient's records. There should be strong password policies as well otherwise accounts could be hacked which might lead to stolen records. Patients should also be able to lock/unlock their records to prevent random(unwarranted) accesses to their records.

A malicious-user(doctor/nurse/sys admin with access to patient records) can copy all patient records onto a storage device such as a pen drive. Hospital employees should be frisked for electronic storage devices while entering and exiting premises to prevent them from downloading patient data onto such devices.

A malicious-user(doctor/nurse/sys admin with access to patient records) can potentially upload all patient records online, thus, internet access should be restricted by using VPNs to prevent uploading of patient data online.

A malicious-user can break into the network to launch more attacks to try and break the system and access the patient records. The network has to be protected using network analyzers and firewalls to detect and prevent intrusion.

A malicious-user can launch malware into the infrastructure to launch further attacks to enable stealing of the patient records. Antivirus has to be regularly updated to prevent attacks by malware.
A malicious-user can potentially use known vulnerabilities in the system dependencies to break into the system and steal patient records. The server has to be protected by frequently patching any third-party library dependencies so that vulnerabilities in them do not propagate to the system. Secure communication should be used by the server using TLS. CSRF tokens should be used for each request made to the server to protect against CSRF attacks.
Enforcing policies are the cheapest form of defense as it requires little work while frisking employees are the costliest as it's a tedious process and care must be taken to not violate the rights of the employees. Other protection mechanisms for the infrastructure have average cost as it requires some form of setup.

Evidence:

Links to online articles supporting above mentioned arguments are given below.

1. Use of Tools to test Network Security (Wireshark, Cain & Abel, tcpdump):
   http://sectools.org/tag/sniffers/
2. Making Employees of Organizations aware of OWASP Top 10
   https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
3. Using Strong Passwords to strengthen account security
   https://www.ted.com/talks/lorrie_faith_cranor_what_s_wrong_with_your_pa_w0rd?language=en
4. Two Factor Authentication helping strengthen security
   https://safenet.gemalto.com/multi-factor-authentication/two-factor-authentication-2fa/

Special Equipment:

- Password Protection: Duo(2 factor authentication)
- Firewall Hardware: SonicWall, WatchGuard
- VPN: OpenVPN
- Antivirus: McAfee, Norton, BitDefender

6. Vulnerability history

1. Cross-Site Request Forgery (CSRF) Vulnerability **CVE-2014-8073**
   i. 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.
   ii. This vulnerability 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 can exploit this vulnerability to perform cross-site scripting attacks, Web cache poisoning, and other malicious activities.
   iii. The NIST CVE page linked on the section title gives details on reporting of this Vulnerability.
   Vulnerability reported by Author: Mahendra at Packet Storm Website (linked below)
   The reference links are
   https://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html
   https://exchange.xforce.ibmcloud.com/vulnerabilities/97692

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

Figure from *https://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html*

   iv. This is a commonly recurring vulnerability Cross-Site Request Forgery (CSRF) (CWE-352), also part of OWASP Top 10 2017 - A8.
   v. OpenMRS can fix this vulnerability by including a unpredictable token in each HTTP request and such token at a minimum must be unique per user session. Also the ideal way to include the token is in a hidden field and not in the URL so that it is not exposed to an attacker.

2. Permissions, Privileges, and Access Control Vulnerability **CVE-2014-8072**
    i. OpenMRS allows a remote attacker to bypass access control restrictions, this is caused by failure to restrict access to the Admin page URL. An attacker could exploit this vulnerability to bypass security restrictions and gain access to the admin module and disrupt the system.
    ii. The NIST CVE page linked on the section title gives details on reporting of this Vulnerability.
    Vulnerability reported by Author: Mahendra at Packet Storm Website (linked below)
    The reference links are
    http://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html
    https://exchange.xforce.ibmcloud.com/vulnerabilities/97693

    iii. This type of vulnerability is a commonly known software design flaw, software with improper access controls. This is along the lines of OWASP Top 10 2017-A4-Broken Access Control and also MITRE Co
    iv. mmon Weakness Enumeration 264 (CWE-264)
    v. This vulnerability is not yet fixed by OpenMRS team, this can be fixed by creating a role based access control list (RBAC) with strict demarcation and associating each user with their roles.

3. Multiple cross-site scripting (XSS) Vulnerabilities **CVE-2014-8071**
    i. OpenMRS standalone 2.1 had multiple XSS vulnerabilities that allowed remote attackers to inject malicious scripts or HTML via the following
        - givenName, familyName, address1, address2 parameters to to registrationapp/registerPatient.page
        - comment parameter to allergyui/allergy.page
        - w10 parameter to htmlformentryui/htmlform/enterHtmlForm/submit.action page
        - HTTP Referer Header to login.htm page
        - returnUrl parameter to the htmlformentryui/htmlform/enterHtmlFormWithStandardUi.page or coreapps/mergeVisits.page
        - visitId parameter to htmlformentryui/htmlform/enterHtmlFormWithSimpleUi.page.

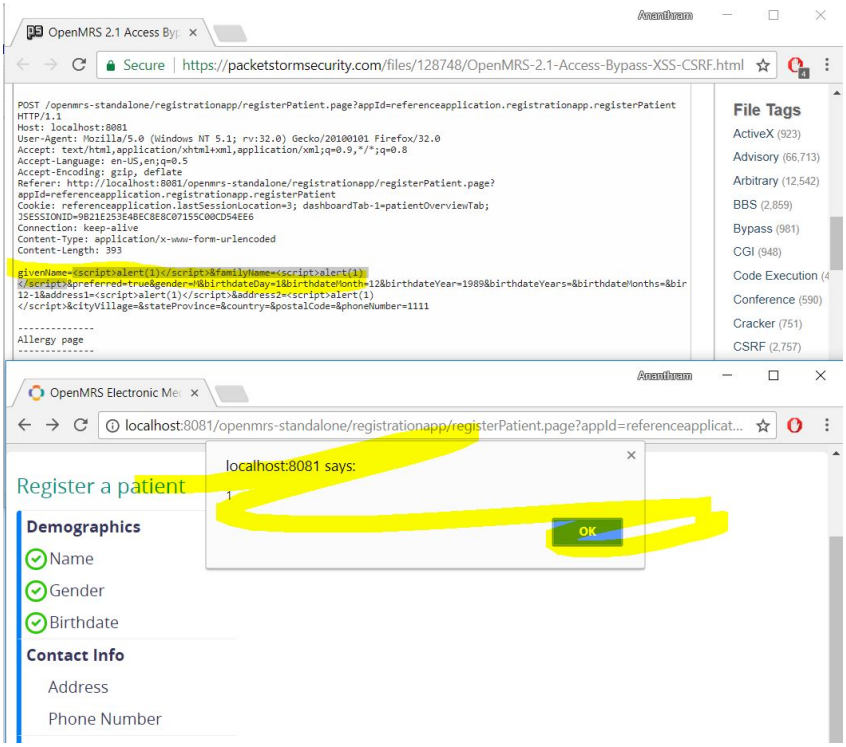
    ii. Vulnerability reported by Author: Mahendra at Packet Storm Website (linked below)
    Sample snippet of vulnerability is below

```
------------------------------------------------------------------------
Multiple Persistent and Reflected Cross-Site Scripting (CVE-2014-8071)
------------------------------------------------------------------------
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
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8081/openmrs-standalone/registrationapp/registerPatient.page?
appId=referenceapplication.registrationapp.registerPatient
Cookie: referenceapplication.lastSessionLocation=3; dashboardTab-1=patientOverviewTab;
JSESSIONID=9B21E253E4BEC8E8C07155C00CD54EE6
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 393

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

Further details and more vulnerability scenarios are illustrated in https://packetstormsecurity.com/files/128748/OpenMRS-2.1-Access-Bypass-XSS-CSRF.html and also https://exchange.xforce.ibmcloud.com/vulnerabilities/97690

iii. This is a commonly recurring vulnerability Cross-Site Request Forgery (CSRF) (CWE-352), also part of OWASP Top 10 2017 - A8.

iv. The vulnerabilities are still not fixed



Our suggestion to fix it is to do proper input validation proper validation in the forms and also for the GET parameters.

4. Cross-Site Request Forgery (CSRF) Vulnerability **CVE-2017-7990**
    i. The Reporting Module 1.12.0 in OpenMRS allows CSRF attacks with resulting XSS attacks, in which administrative authentication is hijacked to insert JavaScript into a name field in webapp/reports/manageReports.jsp
    ii. The exploit is described in the following links
        - https://github.com/openmrs/openmrs-module-reporting/pull/141/commits/0023a659288538d2763835847d3414ecb18b931a#diff-50e25eddc5909110fa3d31090877c2fd
        - https://www.youtube.com/watch?v=pfrIaNvIuFY
    iii. This is a common vulnerability type and is position 8 in Top 10 2017 (A8-Cross-Site Request Forgery (CSRF))
    iv. OpenMRS can fix this vulnerability by including a unpredictable token in each HTTP request and such token at a minimum must be unique per user session. Also the ideal way to include the token is in a hidden field and not in the URL so that it is not exposed to an attacker.