

## Project 3 :

### Part 1: Logging

#### 1. Test Case 1

Session token stored in plain text (true positive)

**Unique ID:** 7.1.1-1

**ASVS:**

7.1.1 Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form.

**CWE:** CWE-532

**URL:** <http://localhost/interface/main/tabs/main.php?token>

**Steps:**

- Login to OpenEMR using valid credentials
- In docker, go to the container image for OpenEMR
- Go to files, and check for file containing access logs
- Search for entries related to the latest login session.
- Check if the logged session tokens are in plain text or hashed format.

**Expected Results:**

- Session tokens should not be present in the logs in plain text. If session tokens are logged, they should be stored in an irreversible, hashed format.

#### 2. Test Case 2

No other sensitive data logged (true positive)

**Unique ID:** 7.1.2-1

**ASVS:**

7.1.2 Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy.

**CWE:** CWE-532

**Steps:**

- Login to OpenEMR using valid credentials
- Create a new patient under Patients tab -> New/Search
- Subsequently, under Patients tab, create a new visit for the patient under Visits tab

**Expected Results:**

- No PII should be logged into the system

### 3. **Test Case 3**

Generic error message with a unique ID shown (true positive)

**Unique ID:** 7.4.1-1

**ASVS:**

7.4.1

Verify that a generic message is shown when an unexpected or security sensitive error occurs, potentially with a unique ID which support personnel can use to investigate

**CWE:** CWE-210

**Steps:**

- Login to OpenEMR using valid credentials
- Go to change password tab
- Generate an input validation error by inputting a password longer than the specified limit

**Expected Inputs:**

- Generic error message with a unique ID to be shown

### 4. **Test Case 4**

Verify that OpenEMR logs all access control decisions, including successful and failed attempts, with relevant metadata for security investigations. (False Positive)

**Unique ID:** 7.1.3-1

**ASVS:**

7.1.3

Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures.

**CWE:** CWE-778

**Steps:**

- Launch OpenEMR using the URL: <http://localhost:80>
- Attempt to log in with invalid credentials.
- Login again with correct credentials via admin and check the logs
- Review the log entries related to the failed login attempt.

**Expected Inputs:**

- The invalid login attempt is logged in

## 5. Test Case 5

Verify that OpenEMR synchronizes its time sources to the correct time and time zone, and preferably logs events in UTC for consistency in global deployments. (False Positive)

**Unique ID:** 7.3.4-1

**ASVS:**

7.3.4

Verify that time sources are synchronized to the correct time and time zone. Strongly consider logging only in UTC if systems are global to assist with post-incident forensic analysis.

**CWE:** CWE-209

### Steps:

- Launch OpenEMR using the URL: <http://localhost:80>
- Login via valid admin credentials
- Go to admin -> system -> logs
- Check the time stamp of the logs

### Expected Inputs:

- OpenEMR records timestamps in UTC for consistency and ease of analysis in global deployments.

## 6. Test Case 6

Verify that OpenEMR log events include sufficient information to enable a detailed investigation of the timeline when an event occurs. (False Positive)

**Unique ID:** 7.1.4-1

**ASVS:**

7.1.4

Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens

**CWE:** CWE-778

### Steps:

- Launch OpenEMR using the URL: <http://localhost:80>
- Log in using admin credentials.
- Perform various actions within the application that generate log entries
- View logs under Admin -> System -> Users

### Expected Inputs:

- The logs should contain all relevant information i.e Timestamp, User, Event Details and error code

## 7. Test Case 7

Verify that OpenEMR logs all access control decisions, including successful and failed attempts, with relevant metadata for security investigations.. (False Positive)

**Unique ID:** 7.2.2-1

**ASVS:**

7.2.2

Verify that all access control decisions can be logged and all failed decisions are logged. This should include requests with relevant metadata needed for security investigations

**CWE:** CWE-285

**Steps:**

- Launch OpenEMR using the URL: http://localhost:80
- Log in using admin credentials.
- Go to Admin -> Users
- Change access control for a user
- Go to Admin -> System -> Logs

**Expected Inputs:**

- The logs should contain all relevant information i.e Timestamp, User, Event Details for update in access control

## 8. Test Case 8

Verify that OpenEMR protects security logs from unauthorized access and modification. (False Positive)

**Unique ID:** 7.3.3-1

**ASVS:**

7.3.3

Verify that security logs are protected from unauthorized access and modification.

**CWE:** CWE-200

**Steps:**

- Launch OpenEMR using the URL: http://localhost:80
- Log in using admin credentials (username: admin, password: pass)
- Navigate to Admin > Users.
- Create a new user with limited privileges.
- Log out of the admin account and log in with the newly created user.

- Attempt to access or modify security logs.

**Expected Inputs:**

- The user should not be able to access the logs

**Adequacy of OpenEMR logging functionality:**

The logging functionality in OpenEMR shows promise in terms of giving thorough information for security investigations and access control monitoring. However, the highlighted gaps in session token storage and PII logging must be addressed immediately to protect user privacy and system security. Implementing secure session token processing and reducing PII logging are critical steps toward increasing OpenEMR's overall logging efficiency. Furthermore, including unique IDs in error messages would greatly improve the system's incident investigation capabilities. The existence of session tokens in plain text in logs poses a severe security concern. This could allow attackers to take over sessions and obtain unauthorized access to the system. OpenEMR should use irreversible hashing or another safe approach to save session tokens in logs. Logging personally identifiable information (PII) may violate privacy laws and reveal critical information. OpenEMR should reduce PII logging and use appropriate data anonymization or redaction procedures. While generic error messages are displayed, the lack of unique identifiers complicates research and troubleshooting efforts.

**Time Taken:**

**Total time taken:** 6 hours

**True Positives found:** 3

**Time taken per true positive:** 2 hours

## Part 2

### Section 1

**List of five chosen attack groups that target healthcare and some techniques used by each:**

1. Fin4
  - 1.1. Application Layer Protocol: Web Protocols
  - 1.2. Command and Scripting Interpreter: Visual Basic
  - 1.3. Email Collection: Remote Email Collection
  - 1.4. Hide Artifacts: Email Hiding Rules
  - 1.5. Input Capture
  - 1.6. Phishing
  - 1.7. Proxy
  - 1.8. User Execution
2. Fox Kitten
  - 2.1. Browser Information Discovery
  - 2.2. Account Discovery: Local Account or Domain Account
  - 2.3. Brute Force
  - 2.4. Credentials from Password Stores: Password Managers
  - 2.5. Data from Cloud Storage
  - 2.6. Valid Accounts
  - 2.7. Create Account: Local Account
  - 2.8. Masquerading: Masquerade Task or Service, Match Legitimate Name or Location
3. LAPSUS\$
  - 3.1. Account Access Removal
  - 3.2. Account Discovery: Domain Account
  - 3.3. Acquire Infrastructure: Virtual Private Server
  - 3.4. Compromise Accounts: Email Accounts
  - 3.5. Create Account: Cloud Account
  - 3.6. Credentials from Password Stores: Credentials from Web Browsers, Password managers
  - 3.7. Impersonation
  - 3.8. Modify Cloud Compute Infrastructure: Create Cloud Instance
  - 3.9. Valid Accounts
4. Leviathan
  - 4.1. Acquire Infrastructure: Domains
  - 4.2. Compromise Accounts: Social Media Accounts/ Email accounts
  - 4.3. Phishing: Spearphishing Attachment or Spearphishing Link
  - 4.4. Valid Accounts
  - 4.5. BITS Jobs
  - 4.6. Establish Accounts: Social Media Accounts or Email Accounts

- 4.7. User Execution: Malicious Link or Malicious File
- 5. EXOTIC LILY
  - 5.1. Acquire Infrastructure: Domains
  - 5.2. Establish Accounts: Social Media Accounts/ Email accounts
  - 5.3. Exploitation for Client Execution
  - 5.4. Phishing: Spearphishing Attachment or Spearphishing Link
  - 5.5. Search Closed Sources
  - 5.6. Web Service
  - 5.7. User Execution: Malicious Link or malicious File

**Table 1:**




Please see the svg in the link below to see the navigator. You must download the svg to see it.

[https://drive.google.com/file/d/13p\\_F82h4kLU69P0I0aUHWzCDOXXvPLMd/view?usp=s\\_haring](https://drive.google.com/file/d/13p_F82h4kLU69P0I0aUHWzCDOXXvPLMd/view?usp=s_haring)

[illegible]

Attack Group	Color
Fin4	Red
Fox Kitten	Yellow
LAPSUS\$	Blue
Leviathan	Orange
Exotic lily	Green
Techniques used by 2 groups	Light Purple



Techniques used by 3 groups	Purple 
Techniques used by 4 groups	Dark Purple 
Techniques used by 5 groups	Grey 

**Table 2:**

Technique Id	Technique	Associated Tactics
T1056	Input Capture	Collection, Credential Access
T1078	Valid Accounts	Defense Evasion, Persistence, Privilege Escalation, Initial Access
T1546	Event Triggered Execution	Privilege Escalation, Persistence
T1098	Account Manipulation	Persistence, Privilege Escalation
T1133	External Remote Services	Persistence, Initial Access
T1547	Boot or Logon Autostart Execution	Persistence, Privilege Escalation
T1197	BITS Jobs	Defense Evasion, Persistence

Below is the same table with tactics as columns.

Technique Id	Technique	Initial Access	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Collection
T1056	Input Capture					X	X
T1078	Valid Accounts	X	X	X	X		
T1546	Event Triggered Execution		X	X			
T1098	Account Manipulation		X	X			
T1133	External Remote Services	X	X				
T1547	Boot or Logon Autostart Execution		X	X			

T1197	BITS Jobs		X		X		
-------	-----------	--	---	--	---	--	--

**Table 3:**

Please open the link below to view table 3 - mitigations to view in google sheets:

[https://docs.google.com/spreadsheets/d/1dJ2\\_KS6tkEUc83FSD-19Kj0YV\\_phopIBdoD5H8H7k3Y/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1dJ2_KS6tkEUc83FSD-19Kj0YV_phopIBdoD5H8H7k3Y/edit?usp=sharing)

Or refer to the below table.

[illegible]



T1098	Account Manipulation					M1032 - Multi-factor Authentication	M1030 - Network Segmentation								
T1197	BITS Jobs					M1037 - Filter Network Traffic		M1028 - Operating System Configuration							
T1547	Boot or Logon Autostart Execution					No mitigations	No mitigations								
T1136	Create Account					M1032 - Multi-factor Authentication									
T1546	Event Triggered Execution					No mitigations	No mitigations								
T1505	Server Software Component					M1047 - Audit									
T1068	Exploitation for Privilege Escalation						M1048 - Application Isolation and Sandboxing								
T1055	Process Injection						M1040 - Behavior Prevention on Endpoint	M1040 - Behavior Prevention on Endpoint							
T1140	Deobfuscate/Decode Files or Information							No mitigations							
T1564	Hide Artifacts							No mitigations							
T1656	Impersonation							M1017 - User Training							
T1036	Masquerading							M1049 - Antivirus/Antimalware							
T1578	Modify Cloud Compute Infrastructure							M1047 - Audit							
T1027	Obfuscated Files or Information							M1047 - Audit							

T1553	Subvert Trust Controls							M1038 - Execution Prevention							
T1218	System Binary Proxy Execution							M1038 - Execution Prevention							
T1110	Brute Force								M1032 - Multi-factor Authentication						
T1555	Credentials from Password Stores								M1027 - Password Policies						
T1056	Input Capture								No mitigations			No mitigations			
T1111	Multi-Factor Authentication Interception								M1017 - User Training						
T1621	Multi-Factor Authentication Request Generation								M1017 - User Training						
T1003	OS Credential Dumping								M1043 - Credential Access Protection						
T1552	Unsecured Credentials								M1047 - Audit						
T1087	Account Discovery									M1028 - Operating System Configuration					
T1217	Browser Information Discovery									No mitigations					
T1083	File and Directory Discovery									No mitigations					
T1046	Network Service Discovery									M1042 - Disable or Remove Feature or Program					

T1069	Permissi on Groups Discover y									No mitigatio ns					
T1012	Query Registry									No mitigatio ns					
T1018	Remote System Discover y									No mitigatio ns					
T1210	Exploitati on of Remote Services										M1050 - Exploit Protectio n				
T1534	Internal Spearphi shing										No mitigatio ns				
T1021	Remote Services										M1032 - Multi-fact or Authenti cation				
T1560	Archive Collecte d Data											M1047 - Audit			
T1530	Data from Cloud Storage											M1047 - Audit			
T1213	Data from Informati on Reposito ries											M1018 - User Account Manage ment			
T1005	Data from Local System											M1057 - Data Loss Preventi on			
T1039	Data from Network Shared Drive											No mitigatio ns			
T1074	Data Staged											No mitigatio ns			
T1114	Email Collectio n											M1041 - Encrypt Sensitive Informati on			
T1071	Applicati on Layer Protocol												M1031 - Network Intrusion Preventi on		
T1105	Ingress Tool Transfer												M1031 - Network Intrusion Preventi on		

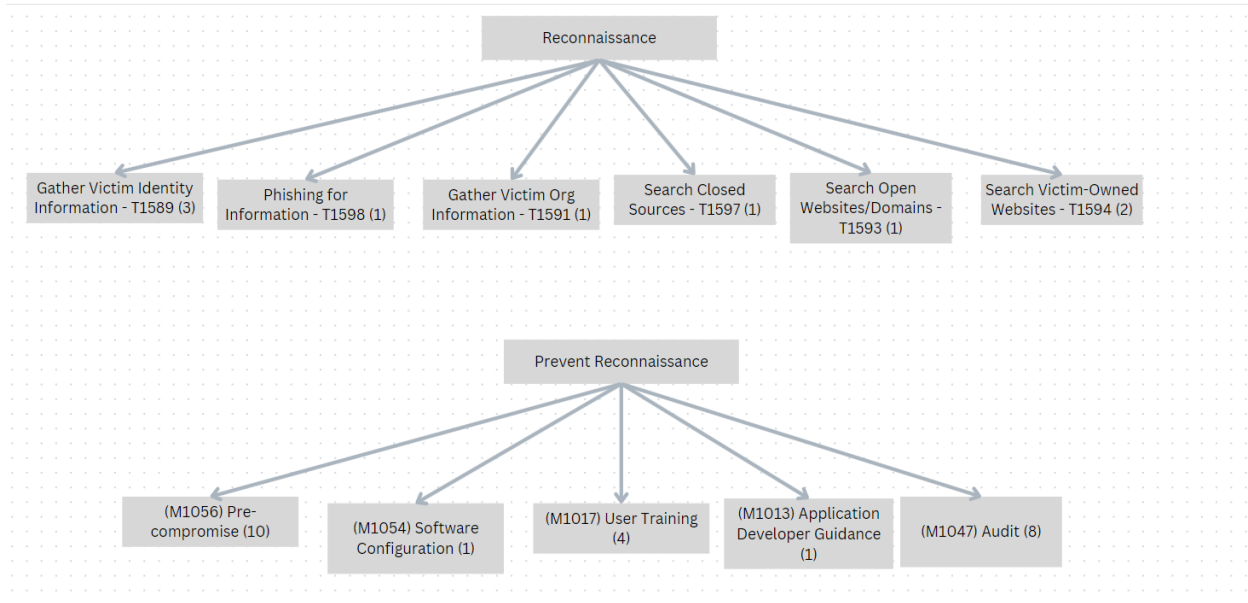


T1572	Protocol Tunneling												M1031 - Network Intrusion Prevention		
T1090	Proxy												M1037 - Filter Network Traffic		
T1102	Web Service												M1031 - Network Intrusion Prevention		
T1041	Exfiltration Over C2 Channel													M1057 - Data Loss Prevention	
T1567	Exfiltration Over Web Service													M1021 - Restrict Web-Based Content	
T1531	Account Access Removal														No mitigations
T1485	Data Destruction														M1053 - Data Backup
T1489	Service Stop														M1022 - Restrict File and Directory Permissions

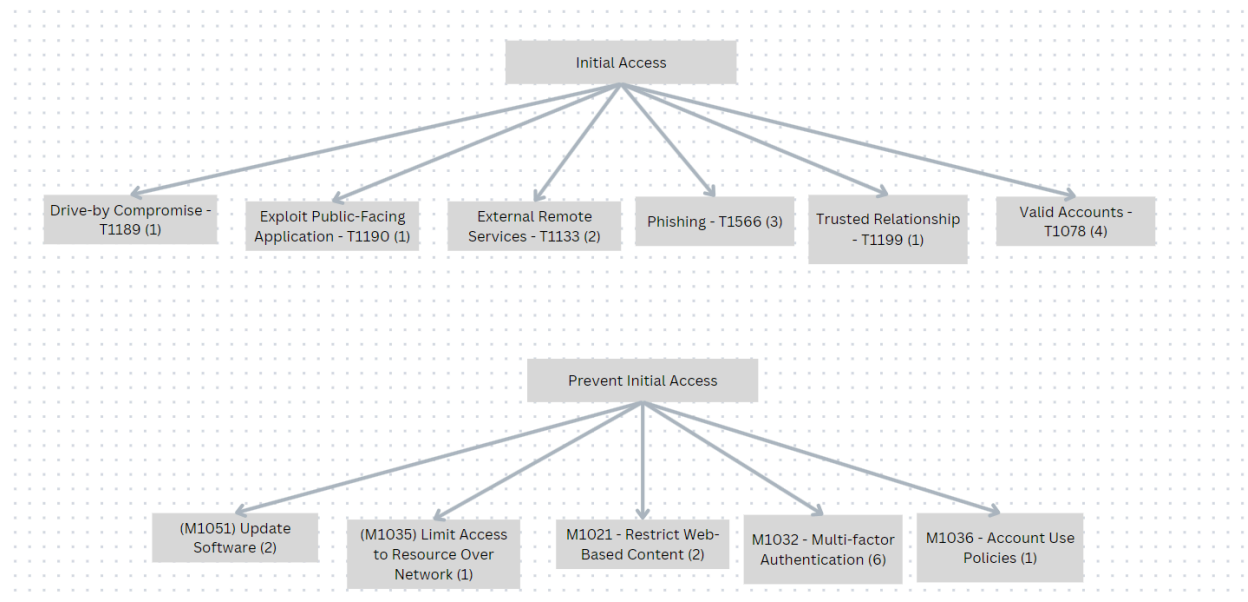
## Part 2 - Section 2

### Attack Trees and Defence trees

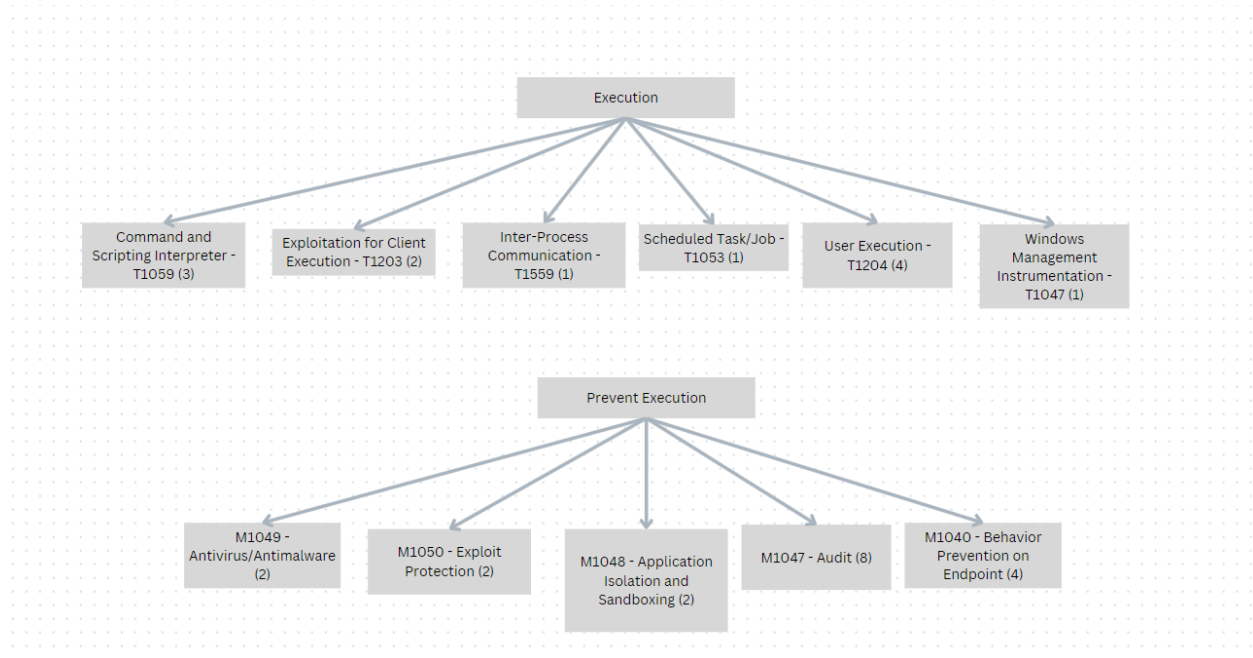
1)



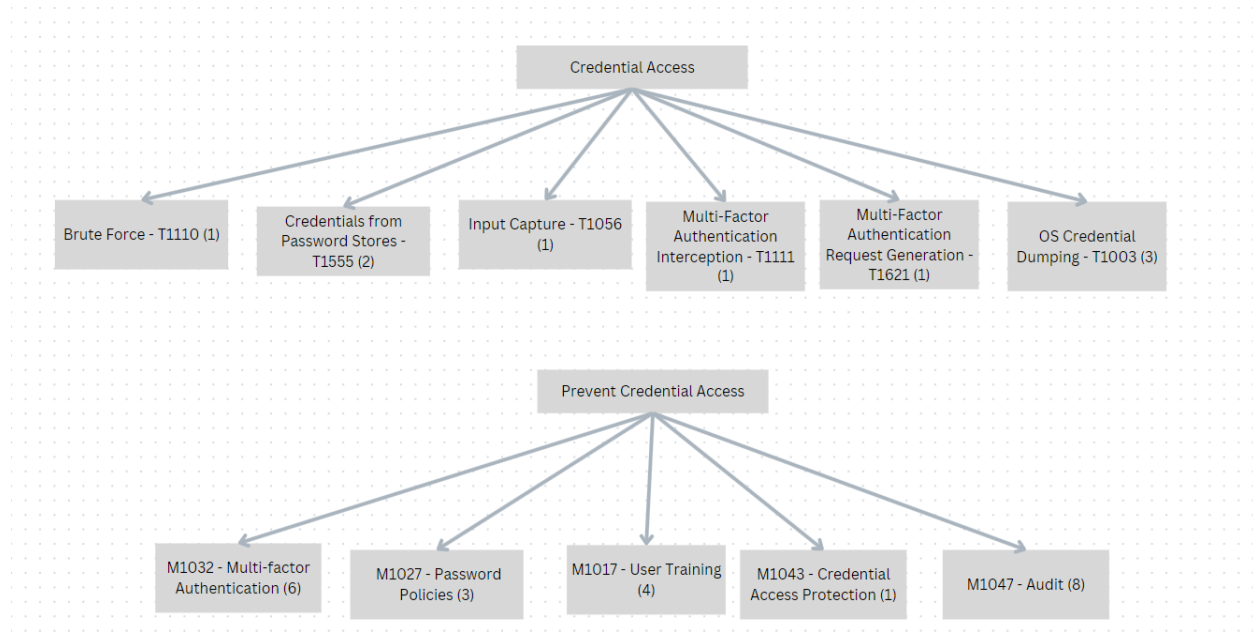
2)



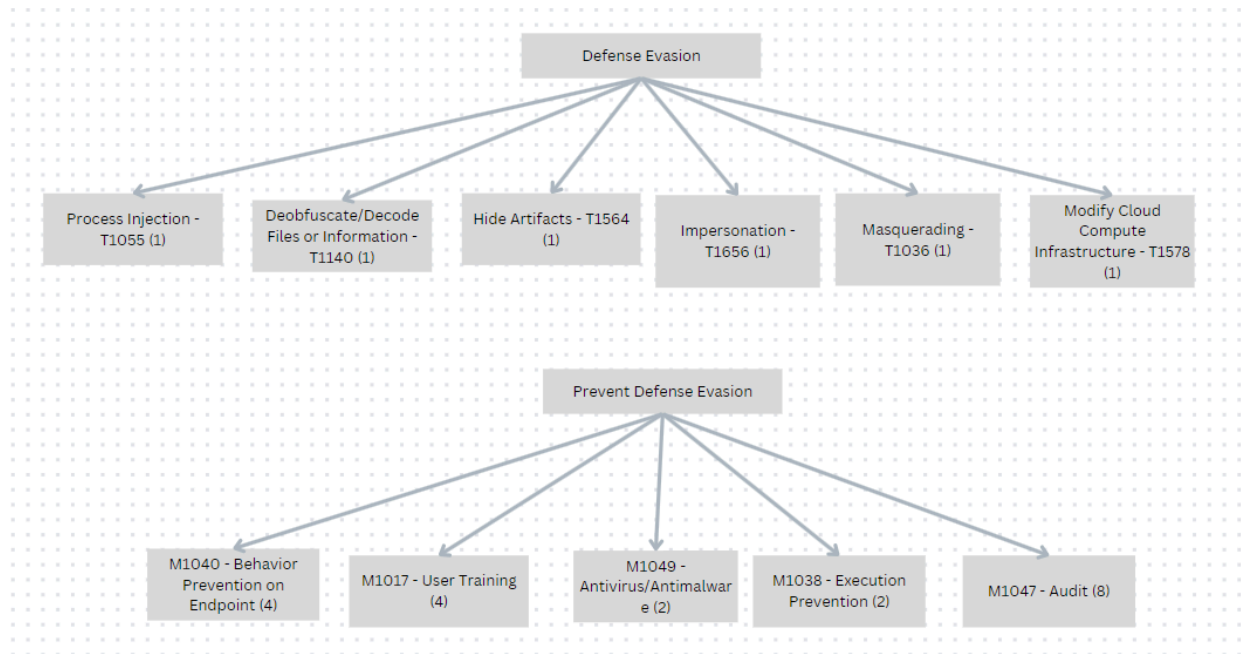
3)



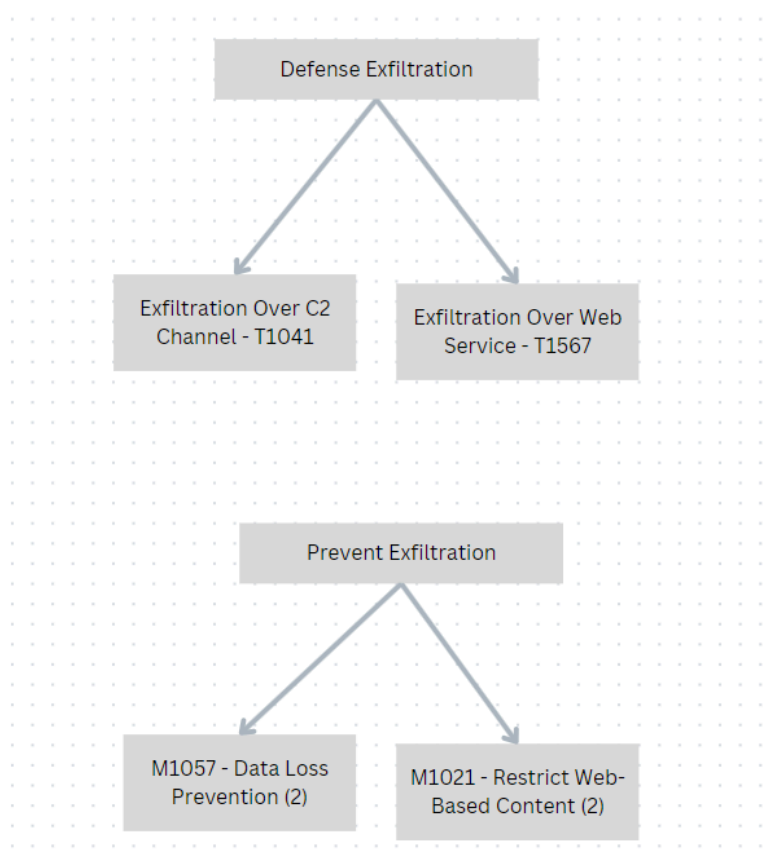
4)



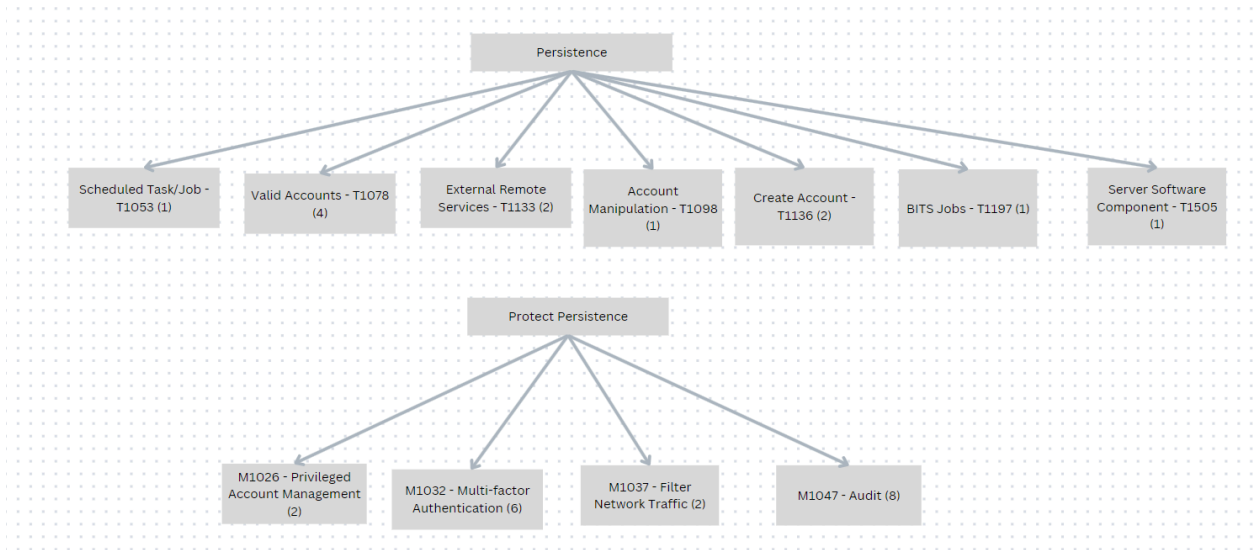
5)



6)

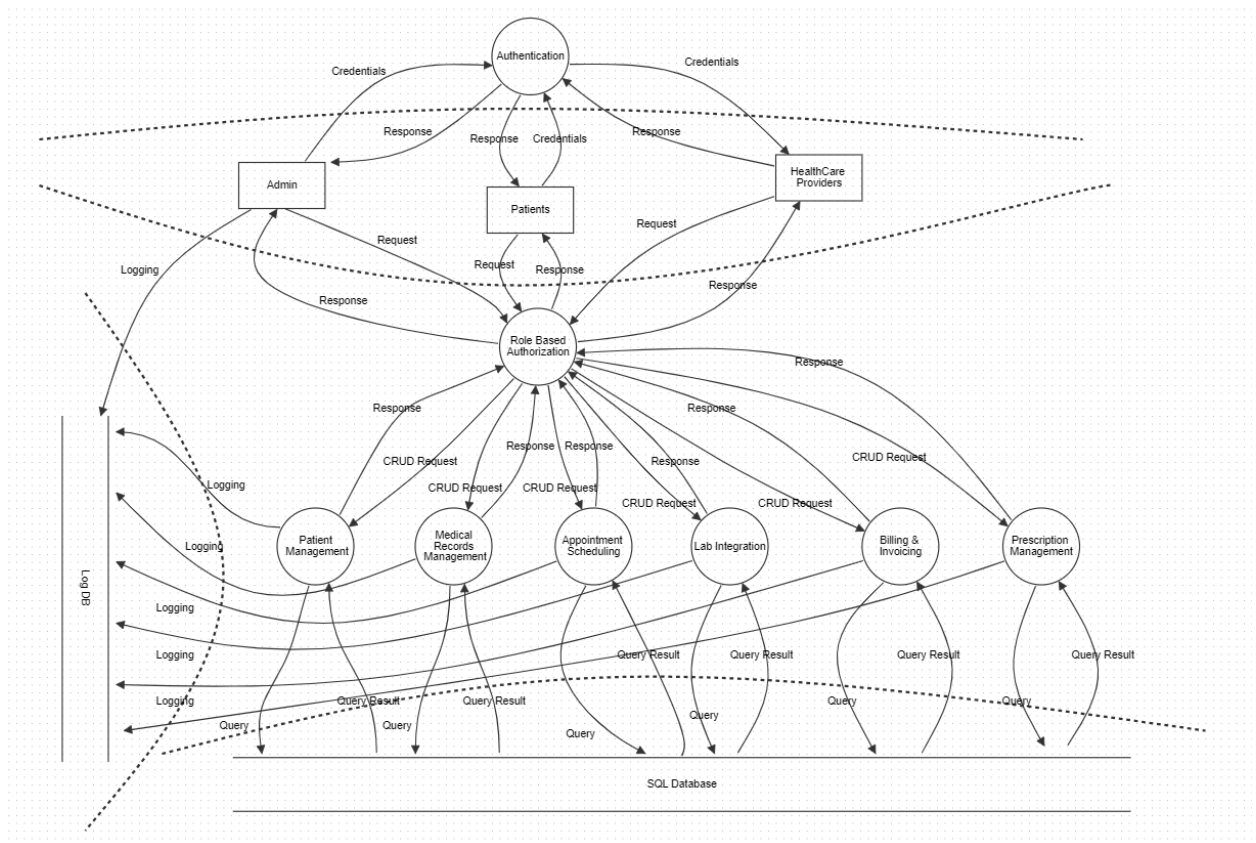


7)



## Part 3: Threat model

### DFD (Threat Dragon)



Threat Dragon Generated Report Link : [Click Here](#)

## **Elevation of Privilege**

### **1. Spoofing (Threat Card K)**

Threat: Your system ships with the default admin password and doesn't force a change.

Mitigation: To prevent attackers from exploiting default admin passwords, systems should not come with preset passwords. Instead, prompt users to create unique, strong passwords at setup. Require password changes at the first login and periodically thereafter to limit attacker opportunities. Enforce password complexity, including length and character variety. OpenEMR should implement strict measures to prompt password changes from default settings, addressing the issue of widely known default passwords.

### **2. Elevation of Privilege (Threat Card 8)**

Threat: An Attacker can enter data that is checked while still under the attacker's control and used later on the other side of the trust boundary.

Mitigation: To counter client-side data manipulation where attackers inject malicious data, ensure robust input validation and sanitization on both client and server sides. Utilize HTTPS for secure data transmission. Conduct frequent security audits and vulnerability tests to fortify application defenses. As observed in previous project parts, client-side bypassing poses significant security risks, emphasizing the need for stringent input handling measures.

### **3. Denial of Service (Threat Card 6)**

Threat: An Attacker can make a server unavailable / unusable without ever authenticating.

Mitigation: To defend against DoS attacks where attackers overload servers with too many requests, adopt rate limiting and traffic filtering to block harmful traffic. Utilize load balancing and scalable infrastructure to manage and distribute incoming requests, lessening DoS attack impacts. Implement monitoring at network and application levels for real-time detection and mitigation of attacks. While OpenEMR allows blocking specific users to prevent authentication, this feature requires manual activation, leaving a gap where DoS attacks can occur.

### **4. Information Disclosure (Threat Card 10)**

Threat: An attacker can read information in files with no ACL's.

Mitigation: To minimize unauthorized access risks, ensure all files have properly configured access control lists (ACLs), adhering to the least privilege principle, granting only necessary permissions.

## **5. Spoofing (Threat Card Q)**

Threat: An Attacker could go after the way credentials are updated or recovered.

Mitigation: Enhance security against credential update or recovery attacks by using HTTPS, multi-factor authentication, and strict identity verification. Conduct security audits and educate users on social engineering. Introduce 2FA for password updates in OpenEMR to strengthen defenses.

## **6. Repudiation (Threat Card 2)**

Threat: An attacker can pass data through the log to attack a log reader and there's no documentation of what sorts of validation are done.

Mitigation: Implement input validation and sanitization to ensure only safe data is logged, masking or omitting sensitive information to safeguard against exploitation.



## **Cornucopia**

### **1. Input Validation Failure (Threat card 7, suit Data Validation & Encoding):**

Threat: Jan can submit crafted payloads that bypass input validation.

Mitigation: Ensuring all data is validated against a strict schema, using whitelisting instead of blacklisting approaches, and canonicalizing data before validation can mitigate this threat.

### **2. Security Misconfiguration (Threat card J, suit Cornucopia):**

Threat: Roman can exploit the application because of insecure default configurations or outdated and vulnerable software.

Mitigation: Regularly update and patch systems, remove unused features, and ensure secure configurations.

### **3. Insecure Deserialization (Threat card 10, suit Cornucopia):**

Threat: Xavier can circumvent application controls because of vulnerabilities in code frameworks, libraries, or components.

Mitigation: Regular vulnerability scanning, using components from trusted sources, and implementing input validation can mitigate this threat.

### **4. Insecure Data Storage (Threat card 8, suit Cryptography):**

Threat: Eoin can access business data because it is not securely encrypted or hashed.

Mitigation: To mitigate this, ensure sensitive data is encrypted both at rest and in transit using strong and current cryptographic practices.

### **5. Authentication Function Exposure (Threat card 2, suit Authentication):**

Threat: James can undertake authentication functions without the user being aware, such as logging in with stolen credentials.

Mitigation: Implementing multi-factor authentication, monitoring and alerting on unusual authentication activities, and educating users on secure credential management.

### **6. Insufficient Transport Layer Protection (Threat card 7, suit Cryptography):**

Threat: Gunter can intercept or modify encrypted data in transit due to weak configurations.

Mitigation: Enforce strong transport layer security (TLS) configurations, use up-to-date protocols and cipher suites, and ensure proper validation of TLS certificates.

## LINDDUN GO

### 1. Identifiability of Inbound Data (I3):

Threat: Data sent to the system can be used to identify the user with a sufficient degree of likelihood.

Mitigation: Limit the collection of identifiable personal data to the minimum necessary and consider transforming the data to a less identifiable form.

### 2. Insufficient Consent Support (U5):

Threat: Consent mechanisms are not properly implemented, leading to personal data being processed without valid consent.

Mitigation: Establish clear and explicit consent collection processes, including easy-to-use options for withdrawal of consent, and ensure data processing aligns with consent given.

### 3. Detectable Communication (D2):

Threat: Communication between the user and the service can be observed, potentially revealing sensitive interactions.

Mitigation: Implement encryption and utilize network security measures like VPNs or Tor to obscure the fact that communication is occurring, especially in contexts where the fact of communication is sensitive.

### 4. No Transparency (U1):

Threat: Lack of transparency in how personal data is collected, processed, and shared can lead to users being unaware of data usage, impacting their ability to make informed decisions.

Mitigation: Provide clear, accessible, and comprehensive information to users about data practices, including how data is collected, used, and shared, as well as the purposes of these activities.

### 5. No User-Friendly Privacy Control (U2):

Threat: If users find it difficult to access and adjust their privacy settings, they may not be able to effectively manage their personal data.

Mitigation: Design intuitive and easily accessible privacy settings interfaces, ensure default settings favor privacy, and provide users with clear guidance on how to adjust their settings.

### 6. Disproportionate Processing (Nc3):

Threat: Data is processed beyond the scope necessary for the original purpose for which it was collected, risking violation of the purpose limitation principle.

Mitigation: Ensure all data processing is strictly necessary for the stated purposes and introduce checks to prevent processing beyond these scopes.

## **Part 4 :**

1. Refer to the below linked Sheet for the test coverage of each chapter.

<https://docs.google.com/spreadsheets/d/1vNqgXnkrT2LyLnRkWT9uUWAPie5aLeAbmJYnVgubQGc/edit?usp=sharing>

**Note :** The test cases for increased converge is mentioned below

### **Chapter 1**

**Test Coverage : 0/42**

**Is increased : No**

### **Chapter 2**

**Test Coverage : 9/57**

**Is increased : Yes**

**Increased Coverage : 11/57**

### **Chapter 3**

**Test Coverage : 4/20**

**Is increased : Yes**

**Increased Coverage : 6/20**

### **Chapter 4**

**Test Coverage : 0/10**

**Is increased : Yes**

**Increased Coverage : 1/10**

### **Chapter 5**

**Test Coverage : 5/30**

**Is increased : Yes**

**Increased Coverage : 6/30**

### **Chapter 6**

**Test Coverage : 0/16**

**Is increased : No**

## **Chapter 7**

**Test Coverage : 6/13**

**Is increased : No**

## **Chapter 8 - Chapter 11**

**These did not have any test cases written pertaining to it. With limited time, prioritization led to insufficient testing of these chapters, affecting overall coverage.**

## **Chapter 12**

**Test Coverage : 1/15**

**Is increased : Yes**

**Increased Coverage : 2/15**

## **Chapter 13- Chapter 14**

**These did not have any test cases written pertaining to it. These chapters have inherently complex requirements that are difficult to test, demanding a deeper understanding and more intricate testing strategies.**

## **2. Black Box Test cases :**

### **Test 1**

**ASVS 4.3.1** Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.

### **CWE 419: Unprotected Primary Channel**

#### **Steps:**

- 1) Go to OpenEMR Login Page and Login into the System as admin.
- 2) Go to Current User Dropdown Menu on the top right corner of landing page
- 3) Select MFA Management Option from the Dropdown
- 4) From the Dropdown of Select/Add New Authentication Method for Administration, select any multi factor authorisation method example TOTP.
- 5) Enter the password asked in the input prompt
- 6) Follow the steps on the screen which says to Install The Google Authenticator app and then by scanning the QR on screen we successfully add the TOTP multi factor authentication mechanism by clicking on the Register button.
- 7) Logout from the Application and once again login into the system

- 8) After logging in using username password we are directed to a page where we have to input the TOTP set in the previous step.
- 9) Add the code from the Google Authenticator app to login in to the system

**Expected Result:** User shall be able to add MFA authentication mechanism and successfully bypass it to login into the system.

## Test 2

**ASVS 3.7.1** Verify the application ensures a full, valid login session or requires re authentication or secondary verification before allowing any sensitive transactions or account modifications.

### CWE Number: 778: Insufficient Logging

#### Steps:

- 1) Go to OpenEMR Login Page and Login into the System as admin
- 2) Go to Current User Dropdown Menu on the top right corner of landing page
- 3) Select MFA Management Option from the Dropdown
- 4) From the Dropdown of Select/Add New Authentication Method for Administration, select any multi factor authorisation method example TOTP.
- 5) Enter the password asked in the input prompt to make sure the sensitive transaction of adding a MFA is authorized by a correct user.

**Expected Result:** Before performing any sensitive transaction such as adding new MFA authentication method system should verify user's authority and identity before allowing the transaction to go through

## Test 3

**ASVS 3.2.1** Verify the application generates a new session token on user authentication.

### CWE-384: Session Fixation

#### Steps:

- 1) Go to OpenEMR Login Page
- 2) Login into the system using correct credentials
- 3) Check the token in URL and note it down
- 4) Logout from the system
- 5) Repeat steps 1,2,3 and check the uniqueness of session token every time user logs in

**Expected Result:** Every time a user logs into the system a new unique session token should be generated every time.

## Test 4

**ASVS 2.8.4** Verify that time-based OTP can be used only once within the validity period

### **CWE-287: Improper Authentication**

#### **Steps:**

- 1) Go to OpenEMR Login Page and Login into the System.
- 2) Go to Current User Dropdown Menu on the top right corner of landing page
- 3) Select MFA Management Option from the Dropdown
- 4) From the Dropdown of Select/Add New Authentication Method for Administration, select any multi factor authorisation method example TOTP.
- 5) Enter the password asked in the input prompt
- 6) Follow the steps on the screen which says to Install The Google Authenticator app by scanning the QR on screen and add the TOTP multi factor authentication mechanism by clicking on Register button
- 7) Logout from the Application and once again login into the system
- 8) After logging in using username password we are directed to a page where we have to input the TOTP set in the previous step. Add the code from the Google Authenticator app to login in to the system and Remember the code.
- 9) Logout from the system
- 10) Try logging in again into the system after 10 mins using the TOTP remembered in step 8
- 11) Invalid code error is shown

**Expected Result:** The stale token shall not allow the user to login in to the system and error should be generated.

## Test 5

**ASVS 5.2.5** Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.

### **CWE- 94: Improper Control of Generation of Code ('Code Injection')**

#### **Attack 1:**

##### **Steps**

- 1) Go to login page for openEmr
- 2) Enter the username as '1' = '1 and password = 1' or '1' = '1
- 3) And try logging in. An invalid input message is displayed

#### **Attack 2:**

##### **Steps**

- 1) On the main page try searching in the top search bar with input '1' = '1 or = 1' or '1' = '1.
  - 2) A page is displayed with message “No matching records found”
- Expected result: The application should return a generic error message like “invalid username/password” or “No matching records” when injection of any form is provided as input. In addition, not providing any details of the error or an error object.

## Test 6

**ASVS 2.1.10** Verify that there are no periodic credential rotation or password history requirements.

### **CWE-263: Password Aging with Long Expiration**

#### **Steps:**

- 1) Go to OpenEMR Login Page and Login into the System.
- 2) Go to Current User Dropdown Menu on the top right corner of landing page
- 3) Select Change Password Option from the Dropdown
- 4) Input current password of logged in user (Make sure current password is not default one and update the default password first using below steps
  - a) Go to OpenEMR Login Page and Login into the System.
  - b) Go to Current User Dropdown Menu on the top right corner of landing page
  - c) Select Change Password Option from the Dropdown
  - d) Input current password of logged in user
  - e) Enter New Password which should be as per the system requirements and specifically of 10 characters only
  - f) Repeat the New Password
  - g) Save changes and update password is shown success )
- 5) Enter New Password which should be as per the system requirements
- 6) Repeat the New Password
- 7) Save changes and update password is shown success
- 8) Repeat steps 2,3,4
- 9) Enter the new password as the password which you have remembered in step 4
- 10) Enter the password of step 9 again in the repeat password input field.
- 11) Error is shown “Reuse of previous password is not allowed”

**Expected Result:** As per ASVS there should not be any password history requirement and the user can set an old password as his/her new current password.

## Test 7

**ASVS 12.1.3** Verify that a file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files, or excessively large files.

### **CWE 770 Allocation of Resources Without Limits or Throttling**

**Attack 1:** Uploading a file of large size Steps:

- 1) Login into openEMR with admin credentials
- 2) On the top header find the "Miscellaneous" tab, and select "new document" option from the tab
- 3) Once you click here you can see a file directory on the left. Select "patient information" (any directory can be selected). An upload page will appear on the right.
- 4) Scroll down where you can see a drag and drop upload file section.
- 5) Upload a 200mb file (downloaded a sample test file from the internet) over here and click the upload button.

**Expected Output:** The application should not allow such large files to be uploaded since it can lead to DoS attacks. It should also properly provide an error handling message making it clear for the user how big a file the user can upload.

**Attack 2:** Uploading large quantity of files Steps:

Follow steps 1-4 from above

- 5) Upload about 800 files into the drag and drop box. (we uploaded about 820 files)

**Expected Result:** Such a huge quantity of files should not be allowed to upload. The page should display an error message stating the same.

**3. Total time (hours and minutes) my team spends to complete the 7 new test cases:**

**Test Planning:** 2 hours and 15 minutes.

**Test Execution:** 4 hours and 30 minutes.

Now, let's recalculate the total time and vulnerabilities per hour:

**1. Total Time Calculation:**

- Test Planning: 2 hours and 15 minutes.
- Test Execution: 4 hours and 30 minutes.
- Total Time: (2 hours + 4 hours) + (15 minutes + 30 minutes) = 6 hours and 45 minutes.
- 45 minutes is 0.75 hours.
- Therefore, Total Time = 6.75 hours.

**2. Vulnerabilities Per Hour Calculation:** The team identified 11 vulnerabilities:

- Total vulnerabilities: 11
- Total time in hours: 6.75 hours.
- Vulnerabilities per hour:  $11 \text{ vulnerabilities} / 6.75 \text{ hours} \approx 1.63 \text{ vulnerabilities per hour}$



4. **Recompute your test coverage- This is mentioned in the sheet as well as answer1 for reference.**

Throughout the testing efforts in Project Parts 1, 2, and 3, our team dedicated a total of 84.75 hours to find true positives. During this time, we identified 49 true positive. This results in a metric of approximately 0.52 true positive vulnerabilities found per hour of total testing effort. This metric is vital for understanding the effectiveness and efficiency of our testing processes across the different project phases.

```
# Calculation for per hour
total_vulnerabilities = 49
total_hours = 84.75
vulnerabilities_per_hour = total_vulnerabilities / total_hours
vulnerabilities_per_hour
```

```
Result
0.57
```

# OpenEmr

**Owner:** Nisarg Prajapati  
**Reviewer:** Nisarg  
**Contributors:**  
**Date Generated:** Fri Mar 29 2024

# Executive Summary

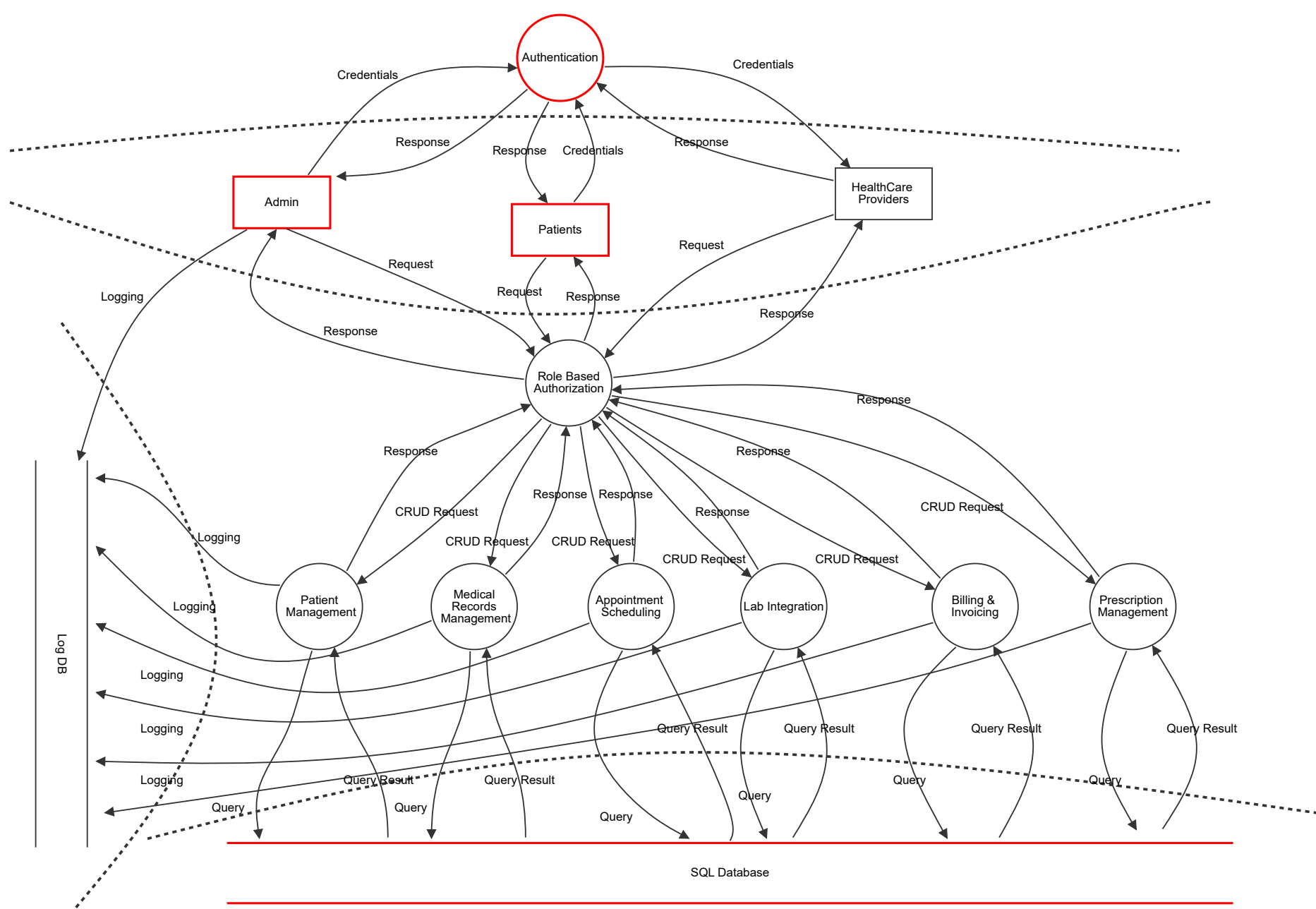
## High level system description

Threat Modelling

## Summary

Total Threats	8
Total Mitigated	0
Not Mitigated	8
Open / High Priority	2
Open / Medium Priority	6
Open / Low Priority	0
Open / Unknown Priority	0

# Stride Open EMR



# Stride Open EMR

## Admin (Actor)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
8	K Spoofing	Spoofing	Medium	Open		The system comes with a default admin password and lacks a mandatory password update feature.	To address the risk of attackers exploiting default admin passwords, developers should implement measures to enforce the creation of unique, strong passwords during initial system setup. Eliminate pre-set default passwords, requiring users to establish their own password upon first use. Mandate periodic password changes and enforce complexity requirements, such as a mix of alphanumeric and special characters, to bolster security. Specifically, in OpenEMR, introduce a feature to prompt users to change the default admin password, which is commonly known and accessible online, to avert unauthorized system access.

## Authentication (Process)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
0	New STRIDE threat	Spoofing	Medium	Open		Provide a description for this threat	Provide remediation for this threat or a reason if status is N/A

## SQL Database (Store)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
5	Tampering of Application with Outdated Packages	Tampering	Medium	Open	6	<p>Outdated packages can expose systems to vulnerabilities.</p> <p>OpenEMR's use of older packages, such as Angular 1.8.3 when the latest version is around 18.0.0, poses a security risk. Updating these packages is essential to enhance security and leverage the latest features and improvements.</p>	An attacker could redirect data flow by exploiting network or protocol weaknesses, like DNS spoofing or hijacking routing protocols. To counter this, organizations should deploy intrusion detection systems, encryption, update protocols regularly, and enforce access controls. Conducting security audits and penetration testing is crucial to uncover and fix vulnerabilities. Additionally, upgrading to the latest software versions can close gaps that older versions might leave open to exploitation.
6	Denial Of Service	Denial of service	High	Open	10	An attacker can disrupt server availability or functionality without needing to authenticate, potentially through methods like DDoS attacks.	To prevent servers from being overwhelmed by Denial of Service (DoS) attacks, implement request rate limiting and malicious traffic filtering. Load balancing and scalable infrastructure can also mitigate attack impact. Continuous network and application monitoring are vital for early detection and response. In OpenEMR, while there is a feature to block user access, it requires manual intervention, creating a brief vulnerability period for potential DoS attacks.
7	Data Oversharing	Information disclosure	Medium	Open	6	Excessive exposure of personal data can have severe repercussions, enabling the misuse of patient information for exploitation.	Minimizing data sharing and conducting frequent audits can help prevent excessive exposure and misuse of personal data.

## Patients (Actor)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
1	User Profiling	Repudiation	Medium	Open	9	Analyzing patterns in data can lead to the unintentional association of unrelated data.	To reduce risks in user profiling, we should enforce rigorous access controls and ethical standards for data use. User data should be used solely for valid reasons and with clear consent to avoid unauthorized profiling.
2	Non Repudiation of Service Usage	Repudiation	High	Open		Identity information can amplify the impact on the system.	By employing robust authentication methods and maintaining comprehensive access logs, we can secure non-repudiation. This creates a verifiable audit trail of user service interactions, eliminating the possibility of users denying their activities.
3	Q-Spoofing	Spoofing	Medium	Open	7	An attacker might target the processes for updating or recovering credentials.	To address vulnerabilities in credential update or recovery processes, developers should adopt secure communication protocols like HTTPS and introduce multi-factor authentication, especially in systems like OpenEMR where it's absent for password updates. Implementing strict identity verification procedures and educating users on social engineering are essential. Incorporating two-factor authentication (2FA) during password updates can significantly bolster security, reducing the risk of unauthorized access. Regular security evaluations are crucial to identify and remediate potential weaknesses proactively.