Testing Techniques To Test OpenMRS System

Ekta Bhatia
Systems and Computer Engineering
Carleton University
Ottawa, Canada
ektabhatia@cmail.carleton.ca

Abstract— **This project report aims to discuss the security and performance vulnerabilities of Open-MRS System –(healthcare management system) by performing Dynamic Performance Analysis and Security Testing. The outcome of the testing performed will help improve the overall performance of medical institutions making use of applications and thereby protect user data. Employing tools such as React Profiler and OWASP ZAP to evaluate the systems frontend under both dynamic and static conditions Thus, overall evaluation of testing techniques and how to integrate it in main framework of the medical application, contributing to enhanced responsiveness, user experience, overall reliability, and security of the system.**

Keywords— Testing, Security, Functionality, Vulnerabilities, Medical Record System, Dynamic Analysis and Penetration Testing, Performance and Authorization Testing.

I. INTRODUCTION

A. OpenMRS System

It is an open-source, Java based electronic medical record system which compromises of numerous repositories(323) which integrate together to work as what we call a system. It aims to work like a black box as it is composed of API's that provide a robust framework for healthcare providers for their operational needs across both developing and developed nations.

It also includes repository for frontend strategy 'openmrs-esm-core' which further involves numerous core components interlinked together with other repositories to form the Open MRS system. In this project we are focusing on performing the security and performance testing under different conditions on the mentioned repository. The frontend system aims to ensure user experience without disrupting the core functionality and allows various modules to interact with each other keeping in mind the extensiveness of the system. Thus ensuring performance and system resilience against vulnerabilities performance is necessary. Various factors leading me to contribute to the system are as follows:

1. Being an open-source project related to ever evolving health industry I wanted to focus on the project, which is flexible, adaptable such that I can perform various testing techniques. Also, this serves a lot of diverse healthcare industries globally it provides me with the flexibility to understand its approach, aim and thus contribute my part to further enhancing it.
2. Also, the system is focused on Heath Information Exchange to facilitate exchange of data hence it would allow me with the opportunity to test the system at peak levels and validate its performance capabilities.
3. It has a decentralized nature of development alongside a lot of technological advancements such as inclusion of Typescript based frontend hence it would further lead me to test for its new capabilities to adapt to changes in version of technology associated with it to ensure its compatibility and performance.
4. Since the project deals with patient data which can be prone to massive data leaks and cybersecurity threats hence one could check the system for its ability to protect patient data from potential vulnerabilities and threats.

II. OBJECTIVE

The main objective of the paper is to check the healthcare system such as Open-MRS System for reliability, security, and performance. To achieve this the system is rigorously tested both in static and dynamic scenarios for performance testing and similarly tested for vulnerabilities such as SQL injections scripting. Thus, the overall testing is focused towards having a trustworthy system with standard performance in all scenarios as required by a healthcare system.

III. METHODOLOGY

To conduct Dynamic Analysis and Penetration Testing and Performance-Authorization Testing, I considered the openmrs-esm-core module which is an essential part of OpenMRS architecture.
The application begins with the login page with username as admin and password Admin123 followed by the home page which is further divided into patient list, service queue and appointment section. Thus, the admin can make use of these components together to register patients, the medical conditions associated with them, their appointment history and so on thereby providing the user with various functionalities about healthcare system. The described sections are further divided into multiple components. The important technologies and frameworks involved in OpenMRS ESM core are:

- Node.js- This provides the environment for developing the scripts which help us perform tasks such as build, development operation.
- Typescript- The chosen system is predominantly written in TypeScript which is the extension of

Javascripts that handles the drawbacks in it better with the help of dynamic variable declaration.

- React-This is one of the most important frameworks used in the system as it brings together the components designed and specified individually to perform together as a system and is a component based livbrary that utilizes a virtual dorm that keeps track of changes and interaction done by user and update ,rerender only those parts of user interface that needs to be changed while keeping UI same thus it is the core advantage of the library. Also, it helps to keep track of the changes in the system and corresponding to it update and render the linked components.
- Restful APIs- It helps to link together the frontend and backend by providing a connection with database - MySQL thereby allowing actions to be taken such as data manipulation.

Next to test the system I considered setting up the chosen application locally on my machine adhering with the vastness of the application. Furthermore, due to complexity of architecture of application ensuring the performance in static and dynamic conditions and checking the system for SQL vulnerability is done by encompassing Dynamic Analysis and Penetration Testing and Performance and Authorization Testing. Overall, it aims to ensure adequate performance security for the end user making use of the system.

A. Dynamic Analysis and Penetration Testing

This technique corresponds to non-functional testing and is a critical technique to figure out the vulnerabilities in the given system under use and ensure the safe use of system by the end user. The technique comprises of two parts as follows:

a) Dynamic Analysis

This refers to the testing of the system and its components in busy state which correspond to the state when the system is in working condition and it begins to analyze the system as follows:

1. System Analysis-It takes in the input of the system running on the localhost as a URL thereby analyse its various aspects such as API request, application hosting platform, application layout and resource usage.
2. Request Validation-After the URL is hosted it begins by injecting invalid inputs, security breach input, bug induced data thereby checking the system response to it such as system validation, acceptance, and rejection criteria and whether the response of system under invalid conditions expected output or otherwise.

Based on the output generated and after its cross verification with expected output and response a report is generated indicating the vulnerability in the system, impact of vulnerability on the system.

b) Penetration Testing

This corresponds to planted testing where we induce vulnerability specifically to check against the web application framework. It begins by following the various aspects as follows:

1. Goal -The initial step is defining the goal which for our system is checking for user-controlled tokens, hosted URL vulnerability and identification of user modified token.
2. Inducing Vulnerability- dummy vulnerabilities are injected such as SQL injection by removing the privileges on local host.
3. Result-The produced output which includes tackling of the system for induced vulnerabilities and exposing the affected elements.

Thus, to perform the testing the Dynamic Analysis and Penetration testing are integrated together, and following test cases are performed by making use of the OWASP ZAP tool.

Test Case:

I. Ensure systems response during static and dynamic penetration testing scenario.
II. Find authorization compromise or security misconfiguration scenario of vulnerability.

Here the static scenario is achieved by the testing during the system at rest or idle state and dynamic scenario is achieved by testing the system by injecting the vulnerabilities while the system is in use.

B. Performance and Authorization Testing

This technique also corresponds to the domain of nonfunctional testing which however is done to check for the performance of the components of the system under test which overall leads to a lacked performance of the system than expected. In general performance testing corresponds to testing the responsiveness and speed of the system. Here combining it with Authorization testing includes testing the system responsiveness and output under the condition when the system is fed with wrong input.

It begins to perform as follows:

1. Initiate Session-After hosting the system in the local host we initiate a session and interact with the system as usual and with wrong inputs as well for authorization testing alongside.
2. Output Analysis- After the initial interaction we end the session of profiling and check the rendering time of each component and analyse the component which took larger time thereby leading to lack of system and also we can keep track of the component that lead it be re-rendered.

To achieve this testing to contribute to performance testing for system and ensure a responsiveness of system for end user especially with a system dealing with healthcare data and service is crucial. For this purpose React Profiler tool is used. The following test scenario are included:

1. Test system response under normal conditions.
2. System resilience and adaptability test during peak and unusual requests.

Here to achieve the first test the system is run under usual with the tool still in session and the second test is to integrate performance and authorization testing such that the invalid input are initiated and then the rendering time

is noticed for the components thereby leading to overall performance lack in the system.

IV. TEST ENVIRONMENT SETUP

To begin with any of the testcases and testing techniques we need to proceed with setting up the systems as follows:

A. Setup Of OPENMRS-ESM-CORE

To conduct the test, we begin with setting up the chosen system i.e. OpenMRS-ESM CORE.The following steps are performed: Installation-OpenMRS system itself is a massive system compromising of more than hundred units and therefore before having to run even one of its subcomponents it required extensive installation of dependent libraries and components. For my chosen system the following installations are made sure of:

Step 1:Install dependencies

- Node.js- It is a runtime environment essential to run JavaScript and can be installed from the official website.
- Npm&Yarn- Npm is installed by default with node.js and yarn can be installed using command prompt.

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ektab>yarn -v
1.22.22

C:\Users\ektab>npm -v
10.5.0
```

Figure 1-Installation verification.

Step2: Clone the repository.
To run the system locally I cloned the openmrs-esm-core in my local machine from the official page.
Step 3: Configuration
The environment variable and path are setup alongside database configuration for any API call.
Step 4: Server initialization
This is achieved by compiling the JavaScript codebase using the following command- 'yarn run: shell.'

```
PS C:\Users\ektab\OneDrive\Desktop\openmrs-frontend\openmrs-esm-core> yarn run:shell
<i> [webpack-dev-server] [HPM] Proxy created: context(path) {
<i>             if (!path) {
<i>               return false;
<i>             }
<i>
<i>             if (path.startsWith(openmrsPublicPath)) {
<i>               if (basename(path).indexOf('.') >= 0) {
<i>                 return true;
<i>               } else {
<i>                 return false;
<i>               }
<i>             }
<i>
<i>             if (path.startsWith(openmrsApiUrl)) {
<i>               return true;
<i>             }
<i>
<i>             return false;
<i>           }  -> https://dev3.openmrs.org/
<i> [webpack-dev-server] Project is running at:
<i> [webpack-dev-server] Loopback: http://localhost:8080/
<i> [webpack-dev-server] On Your Network (IPv4): http://10.0.0.45:8080/
<i> [webpack-dev-server] On Your Network (IPv6): http://[fe80::d05f:1e1:81dc:4c08]:8080/
<i> [webpack-dev-server] Content not from webpack is served from 'src/assets' directory
<i> [webpack-dev-server] 404s will fallback to '/index.html'
<i> [webpack-dev-middleware] wait until bundle finished: /openmrs/spa/
```

Figure 2-Command Run

Verification of application run on http://localhost:8081/ is confirmed and depicted in figure3.

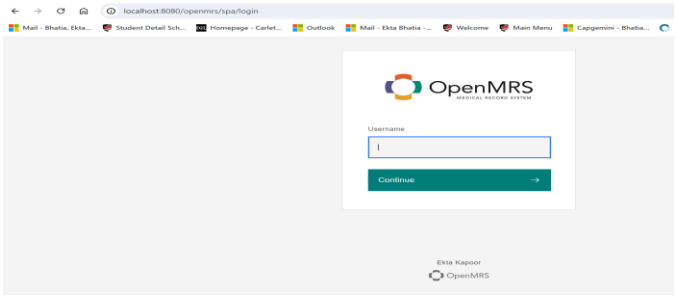


Figure 3

Openmrs system can be logged in using admin username and password as Admin123 as given on the system homepage. Figure 4 demonstrates successful configuration of the OpenMRS system on the local host.

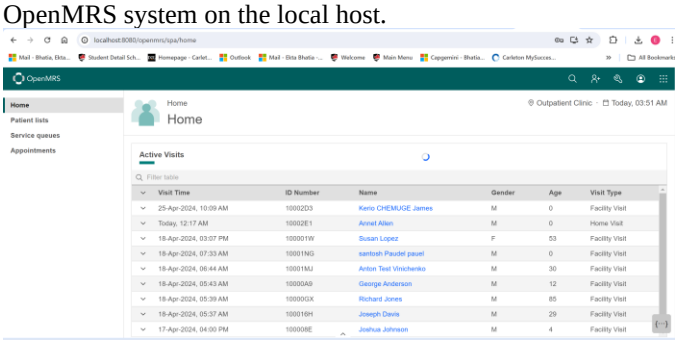


Figure 4

B. Setup Of OWASP-ZAP

To perform Dynamic Analysis and Penetration Testing I made use of OWASP-ZAP tool. To setup in my machine following steps are performed:
Step1: Installation
Download and install it from the official website.
Step 2: Configuration
To configure ZAP to test ZAP is setup as local proxy server by turning on the proxy settings of the browser and initialized to 8081 as demonstrated in figure5.

Figure 5-Proxy settings.

Step3: Session creation
After the proxy settings are updated a new session is created and the local host URL is entered in the sites bar of the zap and the attack is initiated.

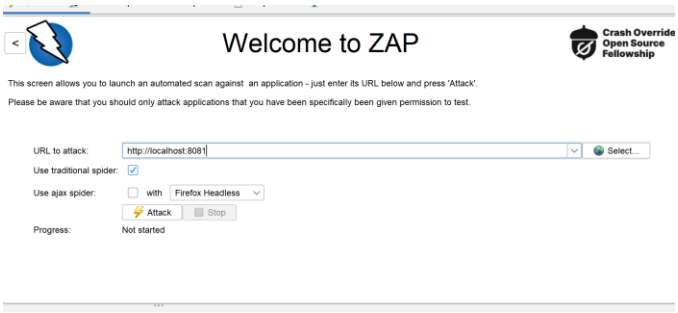


Figure 6

In order to ensure dynamic and static test analysis system attack is considered with initiating the session and without initiating session for the system.

C. REACT PROFILER SETUP

To perform performance and authorization testing the following steps are taken: -
Step1: Confirmation
Confirmation of node.js is installed and extension React Developer Tools extension is installed in browser and is installed from react developer tool extension.
Step2: Profiling
To perform profiling, the record button in profiler section is made use of.
Step3: Testing
After the record button is pressed the profiler records the renders and re-renders of the system traversed.
Step4: Analysis
Flame graphs and ranked charts are used to identify slow rendering components and view of the code is done to verify if the codebase requires rendering of the components as needed.
Figure7 demonstrates all the view of React Profiler

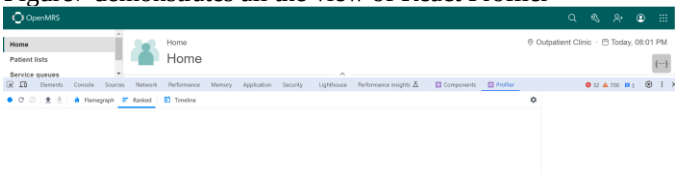


Figure 7

Step 5:The inclusion of onRenderCallback function is performed to retrack the components related to rendering and to do so we add a onRenderCallback function.It enables us to log and handle profiling data.

V. RESULTS.

The section describes the results obtained from the test initiated in the testing technique.

A. Dynamic Analysis and Penetration Testing

After following the steps mentioned in the Environment Setup, I was able to perform the following test and their results are discussed as follows:
Test1: Ensure systems response during static penetration testing scenario.

To achieve this attack is projected on the system without running the system in real time and the following vulnerabilities are configured.
Result: -Figure8 below demonstrates the report addressing site involved, risk level metric(High,medium,low) configured and confidence level of the metric.

Figure 8

Figure9 below reports the reported vulnerability in static condition.

Figure 9

The following two vulnerabilities are reported-

- User Agent Fuzzer-This is a kind of informational reported risk with regard to the output generated when the site is loaded and the response code of the body is compared with the original body which includes hash code. Thus, the error points out to easily decodable code. The prevention can be done by making use of secure hashing method and incorporated encryption algorithms.

- Potential XSS-These analyses the data encoded in HTML tags and checks for if the data is encoded or not when returned in page to prevent execution of malicious data. In order to prevent it following steps can be taken:

1. Output should be encoded-This includes to make sure that the variable are read as text and not code and incorporate AES-128 standard for encryption.
2. Ensure to place the Java Script Context inside a “quoted data value”
3. Use HTML entity encoding to add a variable to a HTML context safely.

Test2: Find authorization compromise or security misconfiguration scenario of vulnerability (Dynamic Penetration Testing.
To perform the testing under dynamic condition we begin by following the various steps as follows:

1. The system is hosted on local host and the URL to it is fed on the ZAP app under the sites section
2. Next the ZAP app scans through the links all the website components and interface in order to first understand and then inject attack

3. As now the testing is under dynamic condition the user is also simultaneously checking through the website and performing action ranging from login to home page, patient registration and appointment viewing followed by scheduling and rescheduling.
4. The screenshot below indicates the dynamic allocation and component visit by ZAP.

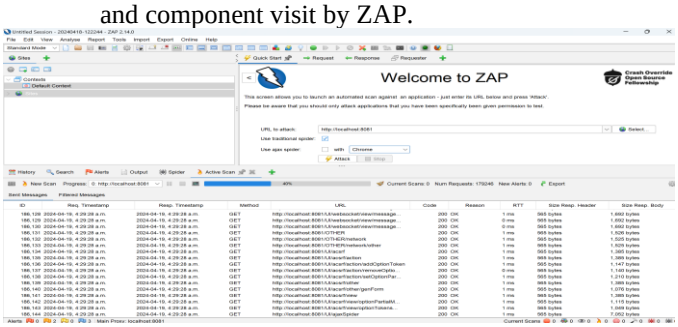


Figure 10

5. After the attack is stopped automatically ,preferred after having visit all the components of the system such that the system is tested under normal and dynamic behavior where we have peak request and processing report is generated for potential vulnerabilities.

Result obtained for potential vulnerabilities depicted in Figure10 and 11.

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	1 (14.3%)	1 (14.3%)	1 (14.3%)	3 (42.9%)
	Low	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Informational	0 (0.0%)	0 (0.0%)	2 (28.6%)	2 (28.6%)	4 (57.1%)
	Total	0 (0.0%)	1 (14.3%)	3 (42.9%)	3 (42.9%)	7 (100%)

Figure 11

Alert type	Risk	Count
.htaccess Information Leak	Medium	669 (9,557.1%)
Absence of Anti-CSRF Tokens	Medium	1338 (19,114.3%)
CSP: Wildcard Directive	Medium	1382 (19,742.9%)
Authentication Request Identified	Informational	2 (28.6%)
Information Disclosure - Sensitive Information in URL	Informational	39 (557.1%)
User Agent Fuzzer	Informational	26028 (371,828.6%)
User Controllable HTML Element Attribute (Potential XSS)	Informational	4134 (59,057.1%)
Total		7

Thus, a total of 7 vulnerabilities are identified in dynamic penetration testing where the memo SQL injection is projected as attack considering the dynamic scenario.

- .htaccess Information Leak-It is identified as medium risk on our system with a analysis count of 669 and it is aimed to alter the configuration of Apache Web Server Software in order to add or disable features provided by Apache Server Software.We can prevent it by ensuring that .htaccess file is not accessible.
- Absence of Anti-CSRF Token-It is identified as medium risk with a count of 1338 of total and happens when a malicious website, email tricks the website in our case OpenMRS system hosted on local host and it gets trapped as the system fails to authenticate the request.It also accounts for XSS(Cross-Site Scripting)attack.

The following steps can be taken to mitigate it.

1. Verify if the system has built in CSRF protection and use it.
2. In case it does not have CSRF protection add tokens to all the components that initiate the change.
3. Include synchronizer token pattern and custom request headers for API-driven site and stateful and stateless software.

- CSP:Wildcard Directive-It is classified as medium risk on our system with a count of 1382 and is also a result of XSS attack and occurs when OWASP ZAP injects malicious script and the system is unable to distinguish between script that is part of application and the script which is injected by third party.

  In order to prevent it we can consider the following measures: -

1. Define allowlist defining the client's accessibility.
2. Define the available directives.
3. Restrict the use of inline code and eval ()
4. Define the policy violation before hosting the system.

- Authentication Request Identified-this Is triggered as a result of the fact that login name and password is common and not unique which can lead to more false negative than false positive.

  To prevent it one can avoid using common name and password, commonly used URL segments, common registration requests.

- Information-Disclosure-Sensitive Information In URL-This is a Information risk identified as a result of enabled debugging message and is returned by web servers such as IIS and Apache. To prevent it one can configure the list of debugging message in the system and set their properties to disable before pushing them on.
- .htaccess Information Leak-It is identified as medium risk on our system with a analysis count of 669 and it is aimed to alter the configuration of Apache Web Server

Software in order to add or disable features provided by Apache Server Software.We can prevent it by ensuring that .htaccess file is not accessible.

- Absence of Anti-CSRF Token-It is identified as medium risk with a count of 1338 of total and happens when a malicious website, email tricks the website in our case OpenMRS system hosted on local host and it gets trapped as the system fails to authenticate the request. It also accounts for XSS(Cross-Site Scripting)attack.

B. Performance and Authorization Testing

According to the methodology described in the previous section we make use of React profiler tool to help us analyse the components rendering and perform the test cases. React profiler is basically an assistive tool for the analysis of performance of web application and makes use of the following tools such as:

1. Component Viewer
2. Profiler
3. Global State Management Viewer

Of which we are considering Profiler to track the rendering of components of system under inspection. To locate the components in our codebase and understand the hierarchical structure we make use of component viewer as well. The following test are conducted, and results are discussed.

Test-1 Test system response under normal conditions.

To perform this test, we generally go through the system without any advance or exceptional out of bound scenario. To begin with we start the system on local host followed by starting a session for admin with correct password in 2 cases by making use of already saved password and other by making use of key stroke to enter password and so on all the functionalities are checked to test the system under normal conditions.

Following results are obtained.

So, I begin with logging in as admin registering a patient under my name.

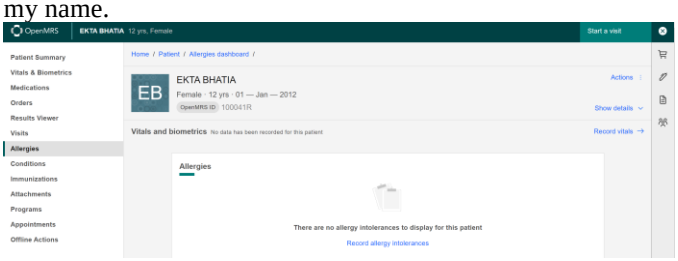


Figure 12

Next the following rendering and re-rendering of system components is obtained.

Case1:I log in as admin and type it by keystroke and make use of pre-saved password then the rendering time is as follows:-

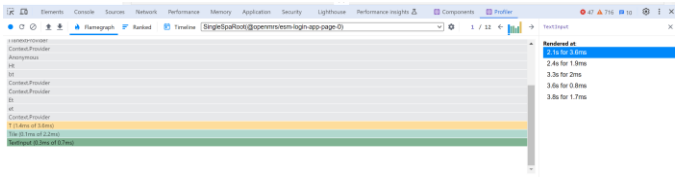


Figure 13-rendering time.

Also, the component and their render and commit time are as described in table1 in static phase where we don't consider any unusual behaviour.

Table 1

Component	Render-time(ms)	Commit time(ms)
Text input	5	2.2
Button	3.8	1.2
Render icon	5.3	4.6

Overall render and commit for the session is as follows:-

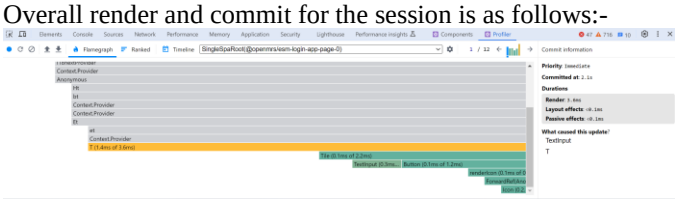


Figure 14

It depicts a render of 3.6 ms overall and commit at 2.1 seconds. Also, the resource utilization and time for processing can be checked by the figure 14 which display the details about space and time.

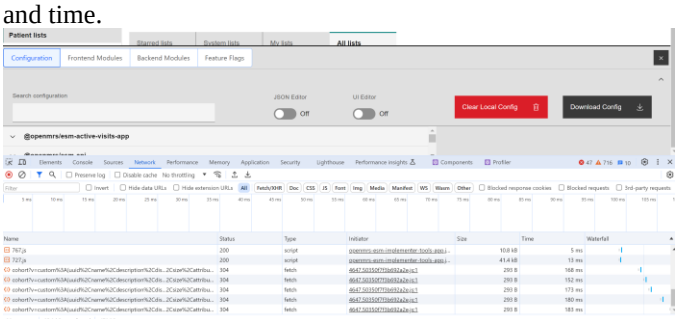


Figure 15-Resource utilization for Test-1

Test-2 Test system under dynamic condition and authorization testing.

To perform this test the system is executed with exceptional scenarios are considered:

1. The admin username is type with wrong password to configure authorization testing.
2. The user is logged in with key stroke password to check in comparison to static conditions followed by normal functioning.
3. The user is logged in with correct details and simultaneous functions are run to check performance under peak setup.

Results obtained are as follows:

In the figure-15 the scenario 1 leads to multiple times the component being rendered

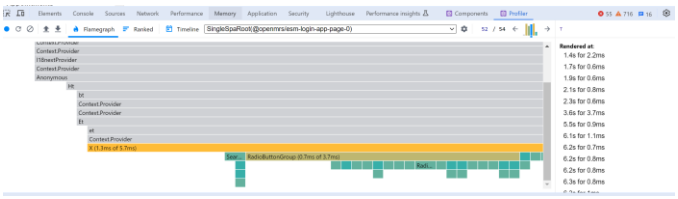


Figure 16-Scenario1.

Table 2

Component	Render-time(ms)	Commit time(ms)
Text input	2.2	1.2
Button	3.9	2.2
Render icon	5.3	4.6

Therefore in this scenario due to multiple inputs for wrong passwords is more but total rendering time is less leading to low commit

In scenario 2 and 3 combined with key stroke and multiple configurations of the system the following rendering time and commit are obtained

Figure 17-Scenario2 and 3 components rendering time and commit.

Table 2

Component	Render-time(ms)	Commit time(ms)
Text input	9	5
Button	0.4	0.2
Render icon	0.1	0.1

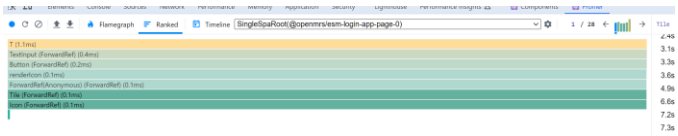
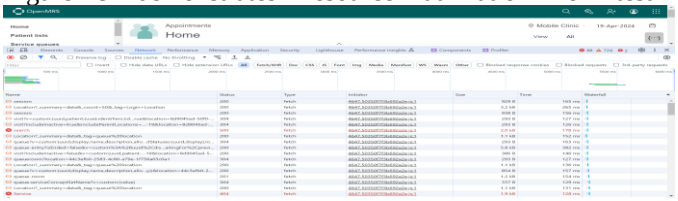


Figure 18-ranked render time of components.

Thus it clearly indicates with dynamic analysis and authorization testing although the components are rendered multiple times and have commit time more than static condition which leads to its drawbacks in performance but is believed to handle multiple request with compromise in performance time. Figure-18 demonstrates resource utilization for test-2



Thus, overall dynamic performance and authorization testing lags than static testing with more resource utilization as well indicated in figure 18.

VI. RESEARCH -QUESTIONS

RQ1: What are the challenges and benefits of incorporating Dynamic Analysis and Penetration Testing and Performance and Authorization Testing in the deployment framework of the project?

Challenges associated with testing techniques are as follows:

- Environment Setup-OpenMRS itself is a huge system designed a decade ago and while working on it to incorporate on my local machine it made me consider version compatibility factors such that to have the system running you need to control the version of dependencies. This clearly indicated although the system is crucial and is massively used it needs to be updated on it version control. Thus in general leads to the drawback of having to need extensive resource setup for the system and also for the tools used to perform the required testing techniques.
- Skill Adequacy-Although I was able to perform the performance and security testing given static and dynamic scenarios but considering the fact of more than thousand of interlinked components would require a skill domain of further injecting attacks in system and thereby need of more computation resource and increase cost.
- False Result Manipulation- Due to the changing nature of environment in which the system is designed and the environment in which the system is tested varies it can lead to unrecognized and unnecessary false positive and thus wasted effort to address the result.

Benefits of incorporating Dynamic Analysis and Penetration Testing and Performance and Authorization Testing in the deployment framework of the project are:

- Vulnerability Assessment-Dynamic Analysis and Penetration testing enables to find risk and vulnerability of the system in real time by injecting attacks thus providing end user confidence by addressing those attacks as mentioned in the result section.
- Performance testing allows us to be aware of the components of the system unnecessarily rendered -re-rendered and thus help the developer make fix to it such that least rendering can be obtained. Such that the system is able to perform up to its required standards.

- Also since the techniques expose overall drawbacks and help address issue in development phase they save the cost and also ensure compliance of system with user experience. This is added advantage to a system such as Open-MRS as it deals with patient

data and any lag in the system or misconfiguration would lead to more extended impact and cost on both developer and user end.

RQ2: Does the deployment of which of above chosen techniques help us address system issue with better resource utilization and response time?

Although the two techniques are extensively different in the domain and specification it specifies I believe in terms of better or more efficient resource utilization Performance and Authorization testing is more prompt ,easily incorporated in our testing pipeline and provide a more holistic and component specific result of the system in less time as compared to Dynamic Analysis and Penetration Testing which has high response time wait and resource utilization capability. Thus the Performance and Authorization testing come in with better capability in the mentioned domain for the developer domain.

RQ3: Comparing both the techniques which of them could also be incorporated with functional testing perspective?

In terms of incorporation with functional testing perspective such as architectural view , subcomponent view I believe the Performance and Authorization testing is better related as although being a non functional testing it heavily make use of architecture components and how are they enclosed as based on that it gets rendered and committed. On the other hand Dynamic Analysis and Penetration Testing does make use of architecture and subcomponent system but that only leads to information risk and the high or medium risk are also based on how the project is mounted on local host, proxy settings and browser configuration which make it subjective to the system it is running on.

VII. CONCLUSION

In conclusion the OpenMRS system is tested using two major techniques Dynamic Analysis and Penetration Testing and Performance and Authorization Testing by locally mounting the system on local host. Further testing is achieved by making use of tools such as OWASP ZAP and React Profiler. The results obtained and drawback and challenges are addressed in the sections of this paper. Future work can be done to explore more injection attack on the system to test for its security in wider domain and also the performance testing can be done with respect to components rendered simultaneously and integrate the tools in CI/CD pipelines such that new changes in the system environment can be checked alongside its deployment allowing enhanced user and developer confidence in the system.

Also I had the opportunity to learn through this project and course how to integrate a industry standard project in my local system, understand their dependencies and code structure to further check for its efficiency and vulnerabilities. Also it gave me an insight on how various testing techniques can be integrated to check the system in hand. It made me observe that a single dependency can lead to a major vulnerability and performance drawback Further maintaining the goal of testing is important as vast projects like that can make the tester to deviate from its goal of finding specific details to an unguided search. I would like to thank my instructor Dr. Nafiseh Kahani for enriching our knowledge about Software testing by maintain a course structure with both theoretical and hands on knowledge perspective.

VIII. REFERENCES

[1] https://github.com/openmrs

[2] https://cwe.mitre.org/data/definitions/352.html

[3] https://web.dev/articles/csp#policy_applies_to_a_wide_variety_of_resources

[4] https://www.zaproxy.org/alerttags/cwe-20/

[5] T. Rangnau, R. v. Buijtenen, F. Fransen and F. Turkmen, "Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines," 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), Eindhoven, Netherlands, 2020, pp. 145-154, doi: 10.1109/EDOC49727.2020.00026. keywords: {Security;Testing;Pipelines;Synthetic aperture sonar;Automation;Application security;Tools;DevSecOps;Dynamic Security Web Testing;Continuous Security;Continuous Integration},

[6] J. Wang and J. Wu, "Research on Performance Automation Testing Technology Based on JMeter," 2019 International Conference on Robots & Intelligent System (ICRIS), Haikou, China, 2019, pp. 55-58, doi: 10.1109/ICRIS.2019.00023. keywords: {Testing;Time factors;System performance;Automation;Tools;Servers;Aggregates;JMeter;Web Performance;Automation Testing;Concurrent},

[7] Z. Huaji and W. Huarui, "Research on web application load testing model," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2017, pp. 1175-1178, doi: 10.1109/ITNEC.2017.8284961. keywords: {Load modeling;Testing;Servers;Time factors;Agriculture;Data models;Browsers;Web Application;Load capacity;performance testing;Load test Model},

[8] Sanjeev Dhavan, "Analysing Performance of Web-Based Metrics for Evaluating Reliability and Maintainability," IEEE, 2008

[9] Yu Yao, Jie Xia, "Analysis and Research on the performance Optimization of Web Application system in high Concurrency Environment", IEEE, 2016

[10] https://jsjungle.dev/mastering-performance-tuning-in-react-with-the-profiler-api/

[11]] Atefeh Tajpour, Suhaimi Ibrahim, and Maslin Masrom. Sql injection detection and prevention techniques. International Journal of Advancements in Computing Technology, 3(7):82–91, 2011.

[12] Owasp. OWASP Foundation — Open Source Foundation for Application Security, 2017.

[13] Ahmad Ghafarian. A hybrid method for detection and prevention of sql injection attacks. In 2017 Computing Conference, pages 833–838. IEEE, 2017

[14]] Zoran. Waptt-web application penetration testing tool. Advances in Electrical and Computer Engineering, 14(1):93–102, 2014.

[15] Dhruv Parashar, Lalit Mohan Sanagavarapu, and Y Raghu Reddy. Sql injection vulnerability identification from text. In 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference), pages 1–5, 2021

[16] A. Javeed, "Performance Optimization Techniques for ReactJS," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-5, doi: 10.1109/ICECCT.2019.8869134. keywords: {Search problems;Rendering (computer graphics);Instruction sets;Time complexity;Servers;Information filters;ReactJS;Performance;stability;multithreading;JavaScript;re-render;Speed;Search},