CSC 515

Software Security Course Project

Team:

Soha Bhatia(sbhatia6)

Akruti Sinha(asinha6)

Shivangi Chopra(schopra4)

Open-source System:

About the system -

OpenEMR: This open-source system provides electronic health care functionality for several countries (including the United States). OpenEMR currently claims to be used to manage over 1 million patient records across the globe.

Link to download:
https://sourceforge.net/projects/openemr/files/OpenEMR%20Current/6.0.0/openemr-6.0.0.zip/download

Version: 6.0.0

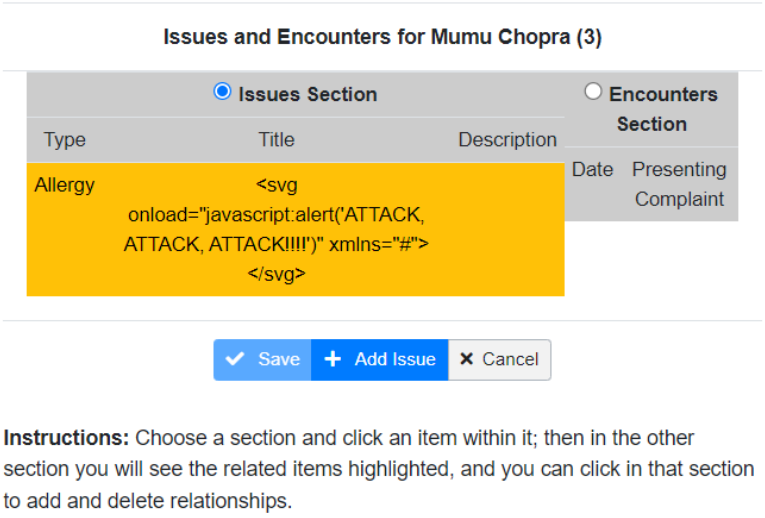
I. Security review on OpenEMR using the 2021 OWASP Top Ten

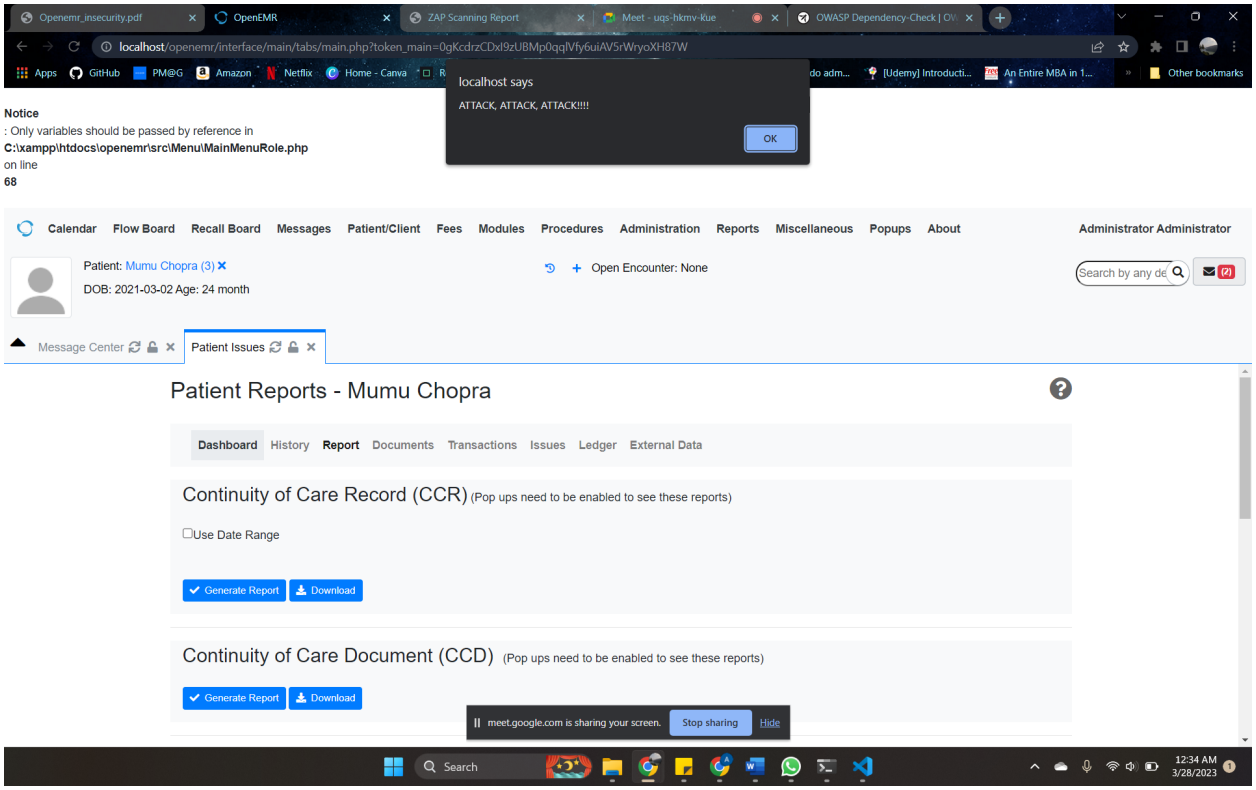
1. A03:2021-Injection : We tested OpenEMR v 6.0.0 application for Cross site request forgery (XSS). The steps which lead to such attack are -

1. Login as administrator to the OpenEMR application through the login page.
2. After login, go to the patients tab and click to create a new patient.
3. The website will navigate to the Medical Record Dashboard screen after adding a new patient.
4. To enter the patient's medical information, click the "edit" button next to the Allergies option. Next select the Add option to enter the patient's medical conditions connected to allergies.
5. Place the provided payload in this title box after selecting the problem type and clicking the save button.

   Payload -
   <svg onload="javascript:alert('ATTACK,ATTACK,ATTACK!!')" xmlns="#"></svg>

6. The attached payload will then be processed under the Reports page of the Dashboard, which presents the patient's information as a report.
7. And it will be displayed as an alert to the user.

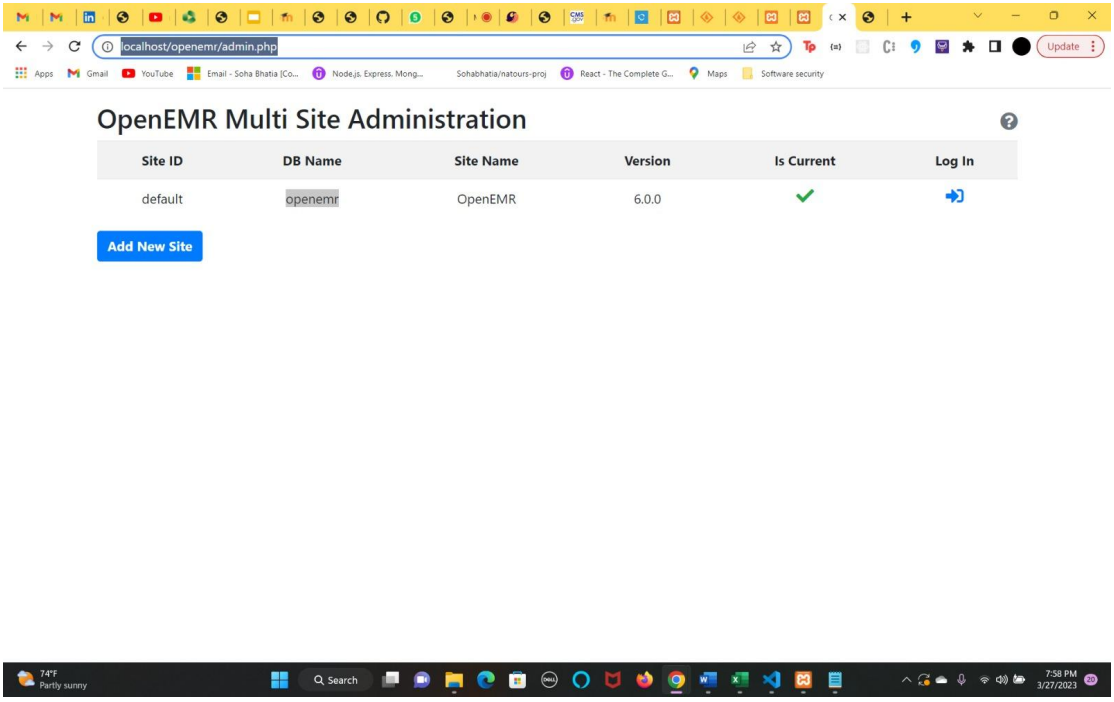




2. A02:2021-Cryptographic Failures: Sensitive information about the applications was easily accessible making the system prone to the sensitive data exposure attack. For doing the attack we followed the following steps:

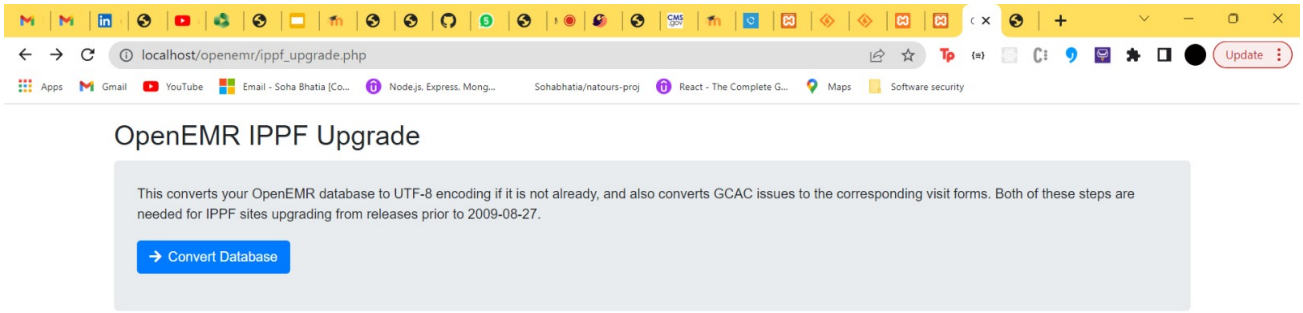
1. Without login-in to the application we tried exploring various endpoints by adding php_file_name.php to the end of the main endpoint.
2. We tried seeing the various php files which were related to admin
3. Finally after visiting **http://localhost/openemr/admin.php**.

   The following content was displayed -



This contains the Database name for the system, which can be used by the attacker to perform other successful SQL attacks on the website.

4. We also tried visiting the URL http://localhost/openemr/ippf_upgrade.php. Which shows the option to convert Database. However, clicking on it throws an error showing the bug has been fixed.

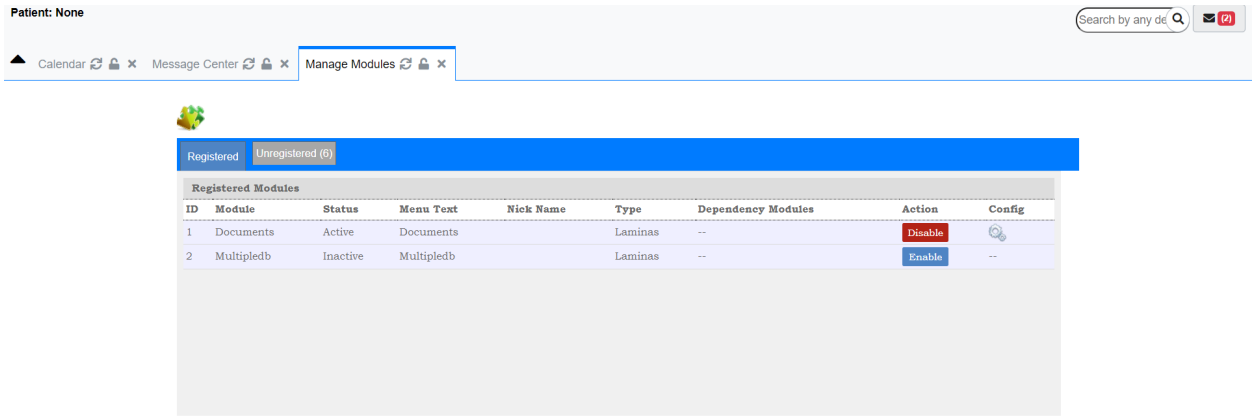


3. A10:2021-Server-Side Request Forgery : We performed a server side request forgery by sending a request to the server containing tampered information by the attacker. By doing so we were able to perform certain action which had only administrative privileges.

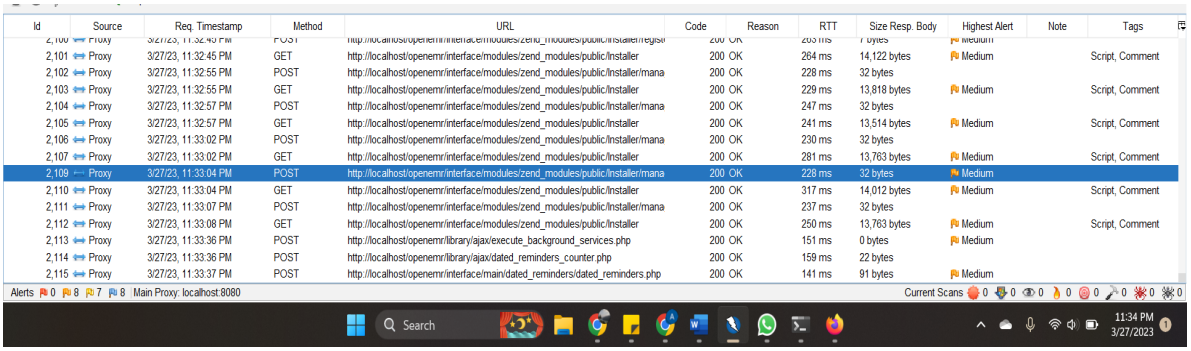
Affected URL :
https://localhost/openemr-6.0.0/interface/modules/zend_modules/public/Installer/manage

The steps involved to perform such an attack are -

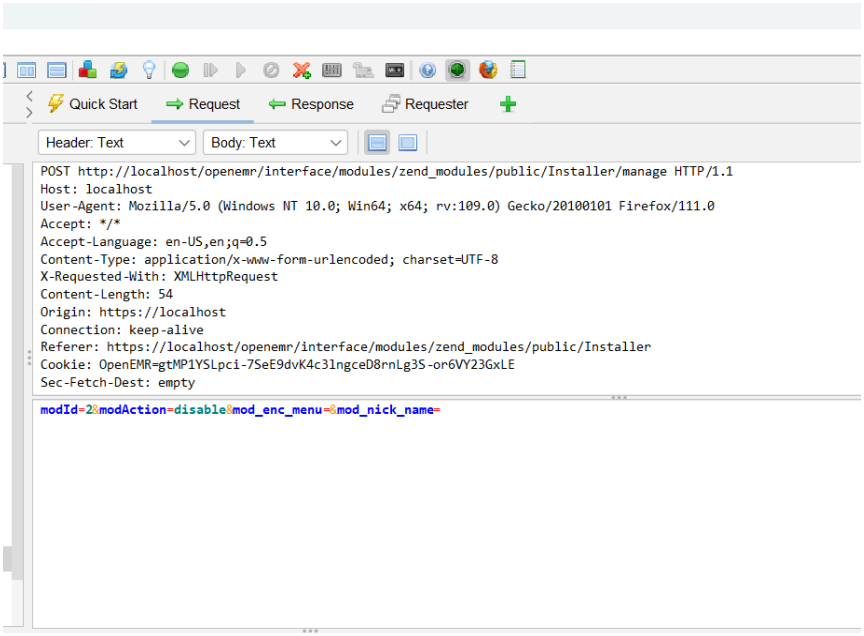
1. An admin user can switch between enabling and disabling registered modules. Login as Admin and we can see the Document Module is Currently Disabled.



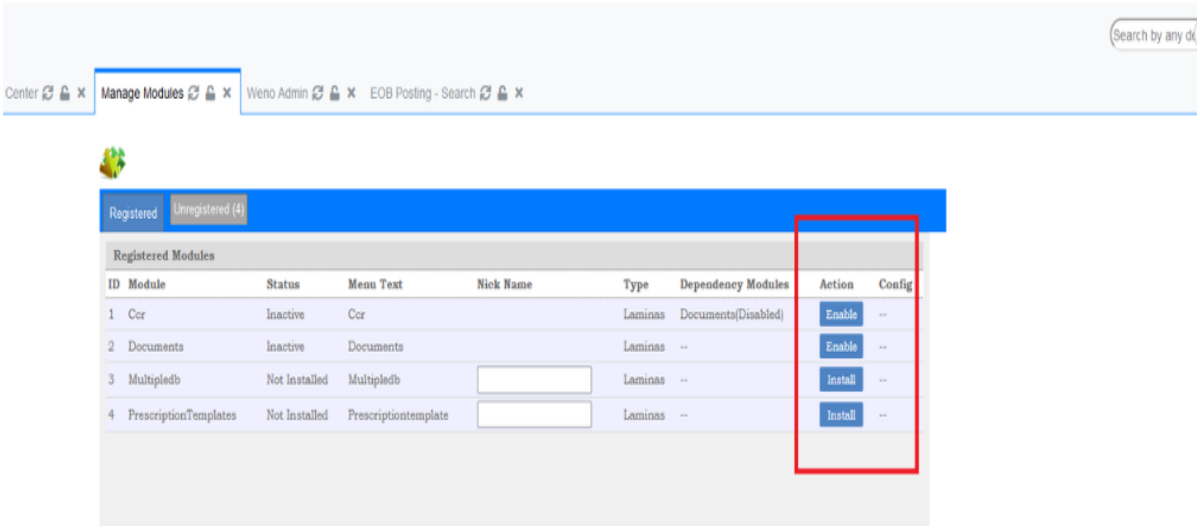
2. Zap was used to record the Admin POST request to the following end point



3. Zap used to capture the request. Non-Privilege User like a physician/accountant modified OpenEMR cookie and modAction Parameter. The OpenEMR cookie was replaced with one from a non-privileged user, like Physician/Accountant, and we were still able to enable and disable modules.

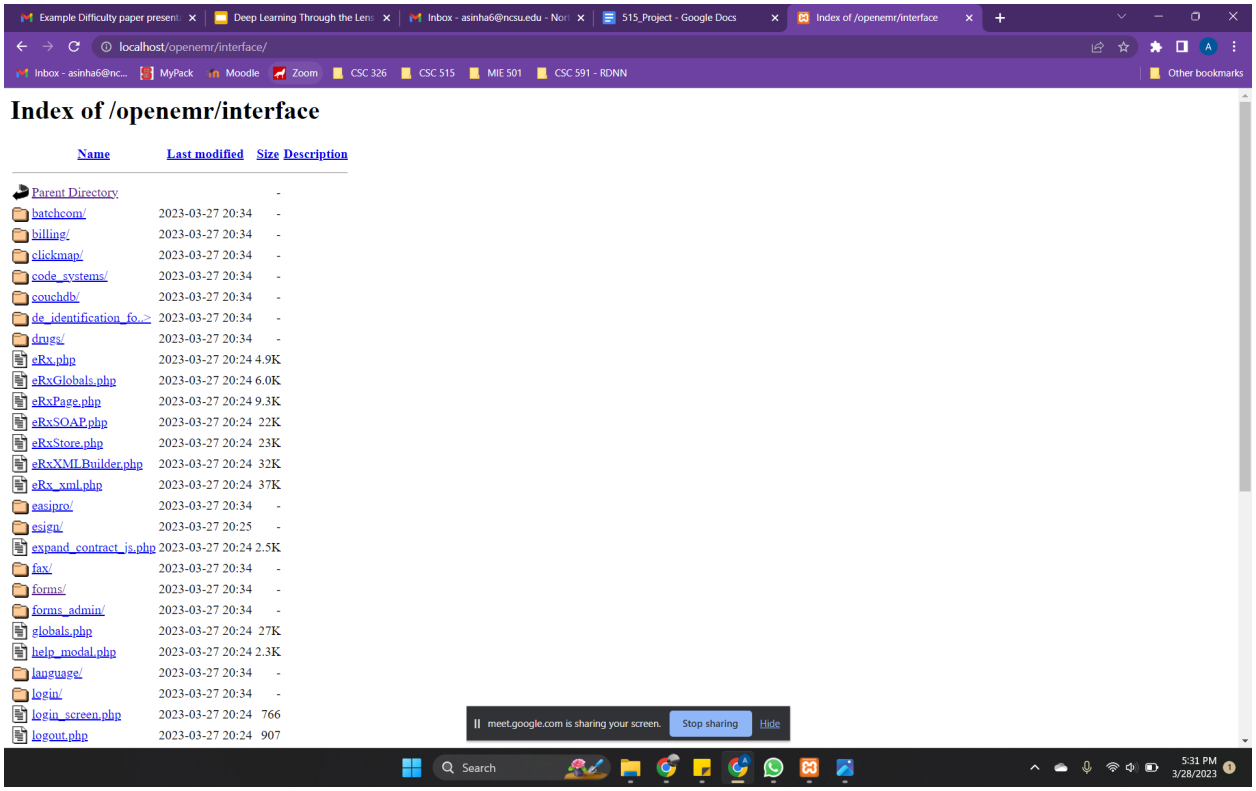


4. Once a non-privileged user tampered with it, the admin account recorded the modules as it was seen.
5. Non-privileged users cannot see the functionality of the module. Nonetheless, we were able to identify the susceptible end-point to make changes to the Module function by intercepting the POST request made by the Admin.

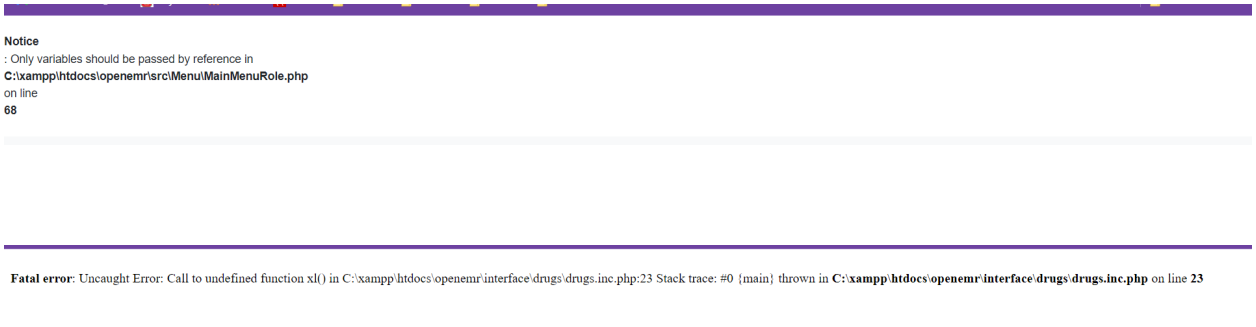


4. A04:2021-Insecure Design : The overall design of this application is very Insecure.
Affected URL: **https://localhost/openemr-6.0.0/interface.**

We are able to view the entire directory structure of this application and also the names of the individual .php files in each directory.



Error stacks are also visible throughout the application which further exposes the folder structure and the components of the code.



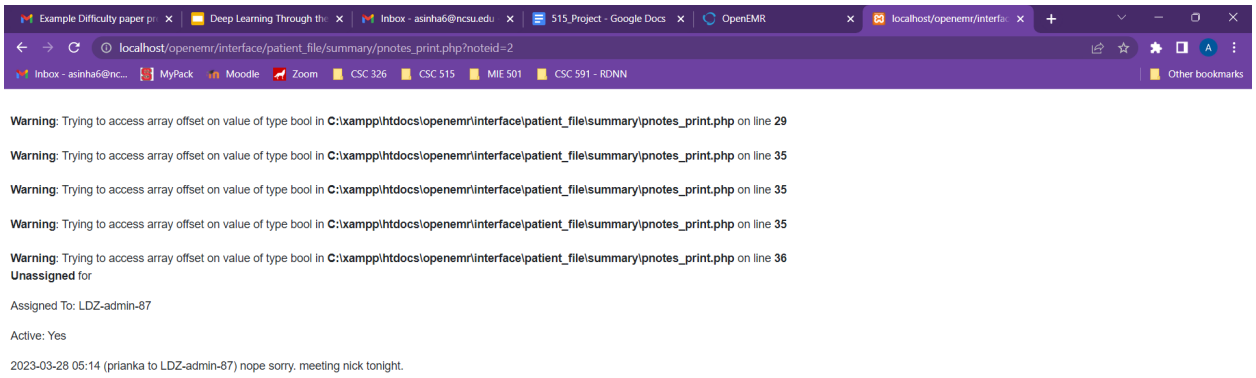
5. A01:2021-Broken Access Control : It is a type of Insecure Direct Object Reference (IDOR) attack. In this attack we are able to access a resource(URL) by changing certain parameters in the query string to view the messages of other people in the application.

Affected URL:
http://localhost/openemr/interface/patient_file/summary/pnotes_print.php?noteid=3

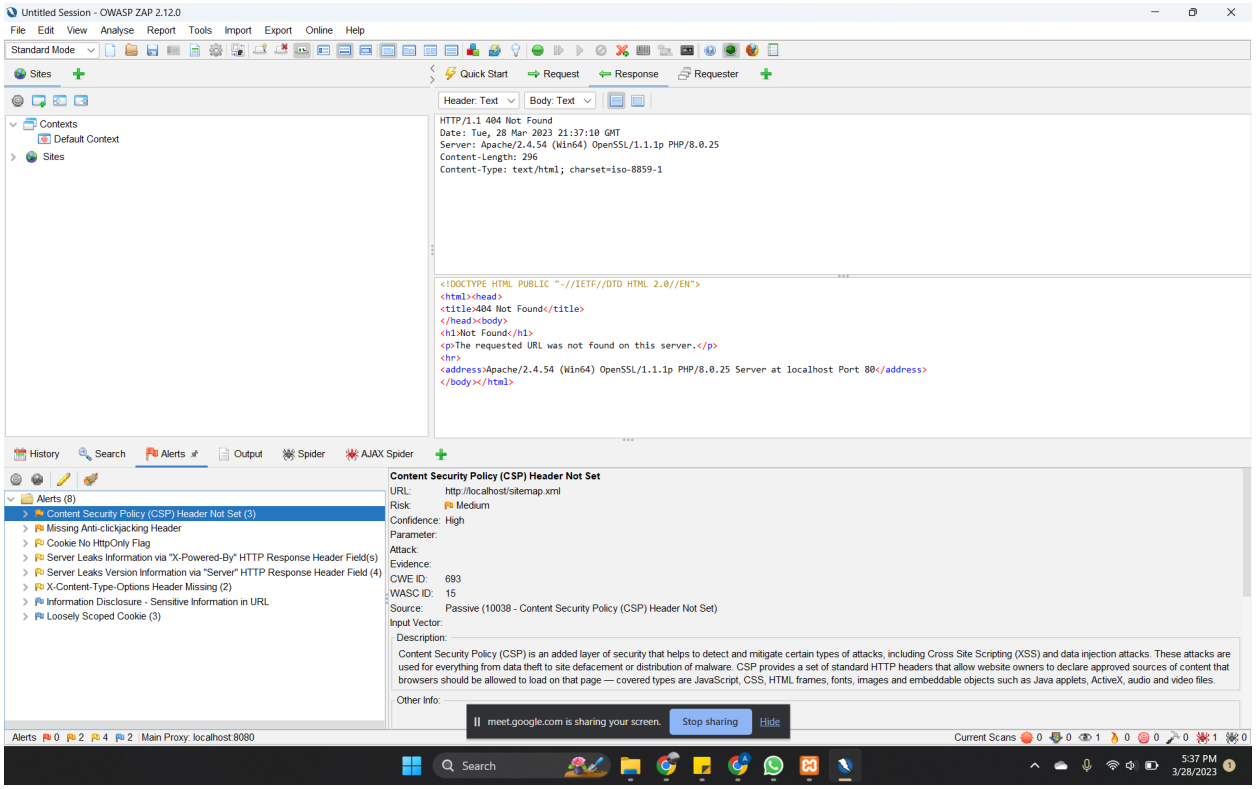
Steps:

1. Login as an admin. And send a message to any other user. Like physician
2. Now, login as a physician and go to view your messages.

3. In the messages tab, click on Print message.
4. This will open a new window in another tab.
5. We tried manipulating the URL by changing the ‘noteid=’ to random numbers.
6. We were able to view the messages of the admin.
7. Further trying any random number we were also able to view messages of any random user of the application which the mentioned noteid in the URL.



6. Using ZAP fuzzer on different endpoints: We tried testing different endpoints against zap fuzzer for SQL injection and XSS attacks by changing the query parameters in search.



However, we couldn't find a successful endpoint. Although it did return with a 200OK status but trying it on the main application did not work.

A detailed ZAP scammer report is attached with this document. That lists down all potential alerts for Target: **http://localhost/openemr/interface/main/tabs/main.php?token_main=hY0WCuvyTGCoRyNhY9qVZHSzFC8BLTxRsOBBobTH**

.

II. Static Analysis and OWASP Dependency Checker

Static Analysis Tool - PHPStan

We used PHPStan, which is a static analysis tool for PHP code that can detect potential issues and provide suggestions for improving the code. It analyzed the code without actually executing it and thus is a reliable testing method. PHPStan looked for problems like type errors, undefined variables, missing method calls, and incorrect function arguments.
It then produced a report (attached) summarizing the findings of its static study of a PHP codebase. The report detailed the number and kinds of issues discovered, as well as recommendations for how to resolve them.

OWASP Dependency Checker:

The OWASP Dependency Checker is a software composition analysis (SCA) tool that assists in identifying and tracking third-party components used in an application, as well as the vulnerabilities connected with them. It helped in scanning the dependencies of the project and produced a report with information about any known vulnerabilities or problems in those dependencies.

Both the tools generated an analysis report for our OpenEMR project. Based on Various factors like issue severity,CVSS score, CV count and dependency in the project, the following top 5 vulnerabilities were identified and some suggestions are made for the same:

1. **Issue:** The package **underscore** from 1.13.0-0 and before 1.13.0-2, from 1.3.2 and before 1.12.1 are vulnerable to Arbitrary Code Injection via the template function, particularly when a variable property is passed as an argument as it is not sanitized

   Suggested Change:
   - Upgrade underscore to version 1.13.0-2, 1.12.1 or higher.
   - Remove Unwanted dependencies.
   - Monitor the project's dependencies on a regular basis for any new vulnerabilities or issues that may emerge. Consider creating alerts or notifications for new vulnerabilities or problems linked to the project's dependencies.

Reference:

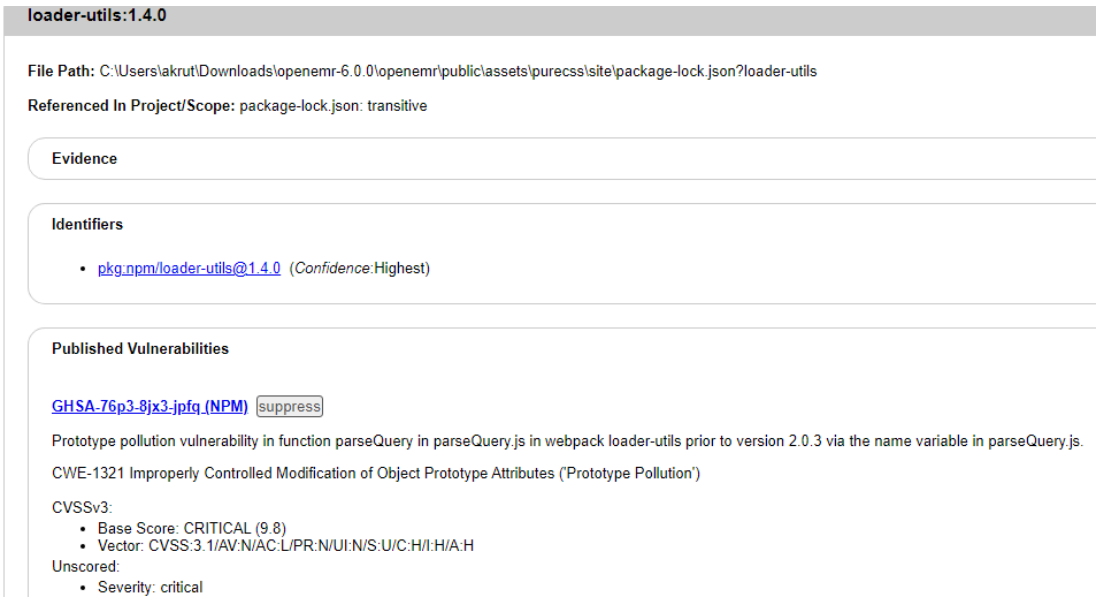
CVE Reference: https://nvd.nist.gov/vuln/detail/CVE-2021-23358

2. **Issue:** A major vulnerability was found in package loader-utils.
It exposes the risk of Prototype pollution vulnerability in function parseQuery and Regular expression denial of service (ReDoS) flaw in Function interpolateName. The vulnerability exists due to the insecure regex pattern used for the `resourcePath` variable in `interpolateName.js`, allowing an attacker to crash the application by providing a malicious input.

Suggested Change:

- change loader-utils version to 3.2.1. This issue has been patched in versions 1.4.2, 2.0.4 and 3.2.1, for Angular you use 3.2.1.
- Remove Unwanted dependencies.
- Monitor the project's dependencies on a regular basis for any new vulnerabilities or issues that may emerge. Consider creating alerts or notifications for new vulnerabilities or problems linked to the project's dependencies.

Reference:



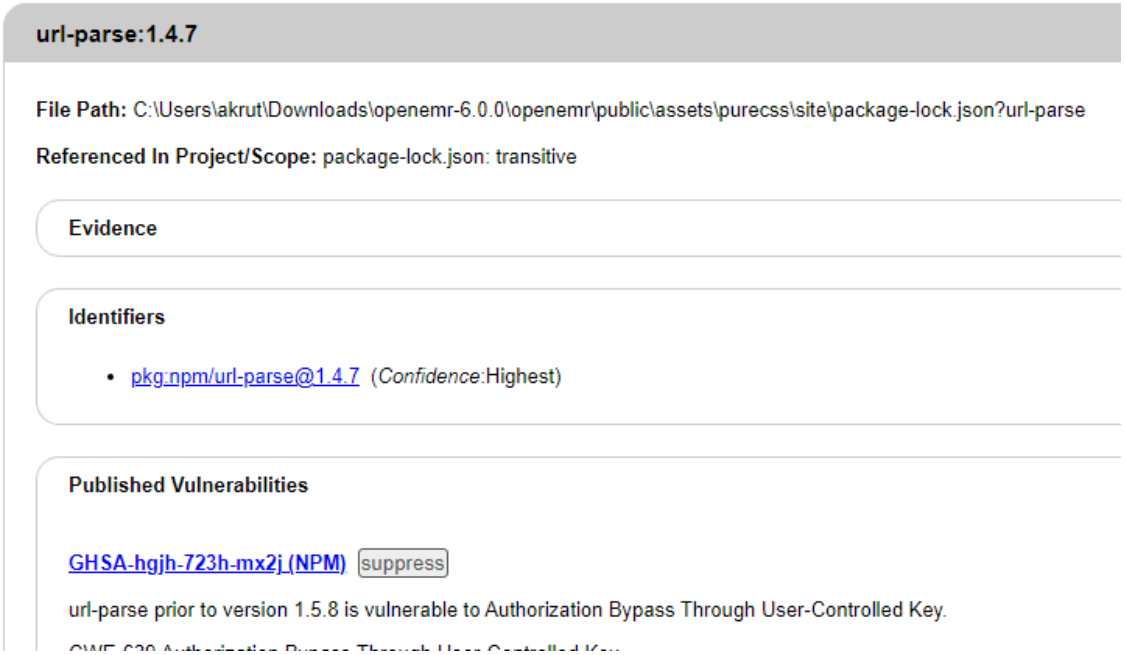
CVE Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-37599

3. **Issue:** Project is using url-parse 1.4.7 Url-parse is a Small footprint URL parser that works seamlessly across Node.js and browser environments. Affected versions of this package are vulnerable to Authorization Bypass Through User-Controlled Key due to incorrect conversion of @ in the protocol field of the HREF.

Suggested Change:

- Upgrade url-parse to version 1.5.2 or higher.
- Remove Unwanted dependencies.
- Monitor the project's dependencies on a regular basis for any new vulnerabilities or issues that may emerge. Consider creating alerts or notifications for new vulnerabilities or problems linked to the project's dependencies.

Reference:



4. **Issue:** Multiple function definitions/ classes not found in static analysis
   Suggested Change:

   - Review the reported issues.
   - Remove deprecated function calls/add necessary function definitions.
   - Test the changes made.

Reference:

```
PHPStan\Analyser\Error::__set_state(array(
   'message' => 'Function attr not found.',
   'file' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\billing
   'line' => 992,
   'canBeIgnored' => true,
   'filePath' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\bil
   'traitFilePath' => NULL,
   'tip' => 'Learn more at https://phpstan.org/user-guide/discovering-symbols',
   'nodeLine' => 992,
   'nodeType' => 'PhpParser\\Node\\Expr\\FuncCall',
   'identifier' => NULL,
   'metadata' =>
  array (
  ),
)),
143 =>
PHPStan\Analyser\Error::__set_state(array(
   'message' => 'Function getPatientData not found.',
   'file' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\billing
   'line' => 994,
   'canBeIgnored' => true,
   'filePath' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\bil
   'traitFilePath' => NULL,
   'tip' => 'Learn more at https://phpstan.org/user-guide/discovering-symbols',
   'nodeLine' => 994,
   'nodeType' => 'PhpParser\\Node\\Expr\\FuncCall',
   'identifier' => NULL,
   'metadata' =>
  array (
  ),
)),
```

5. **Issue:** Insecure coding vulnerabilities and errors

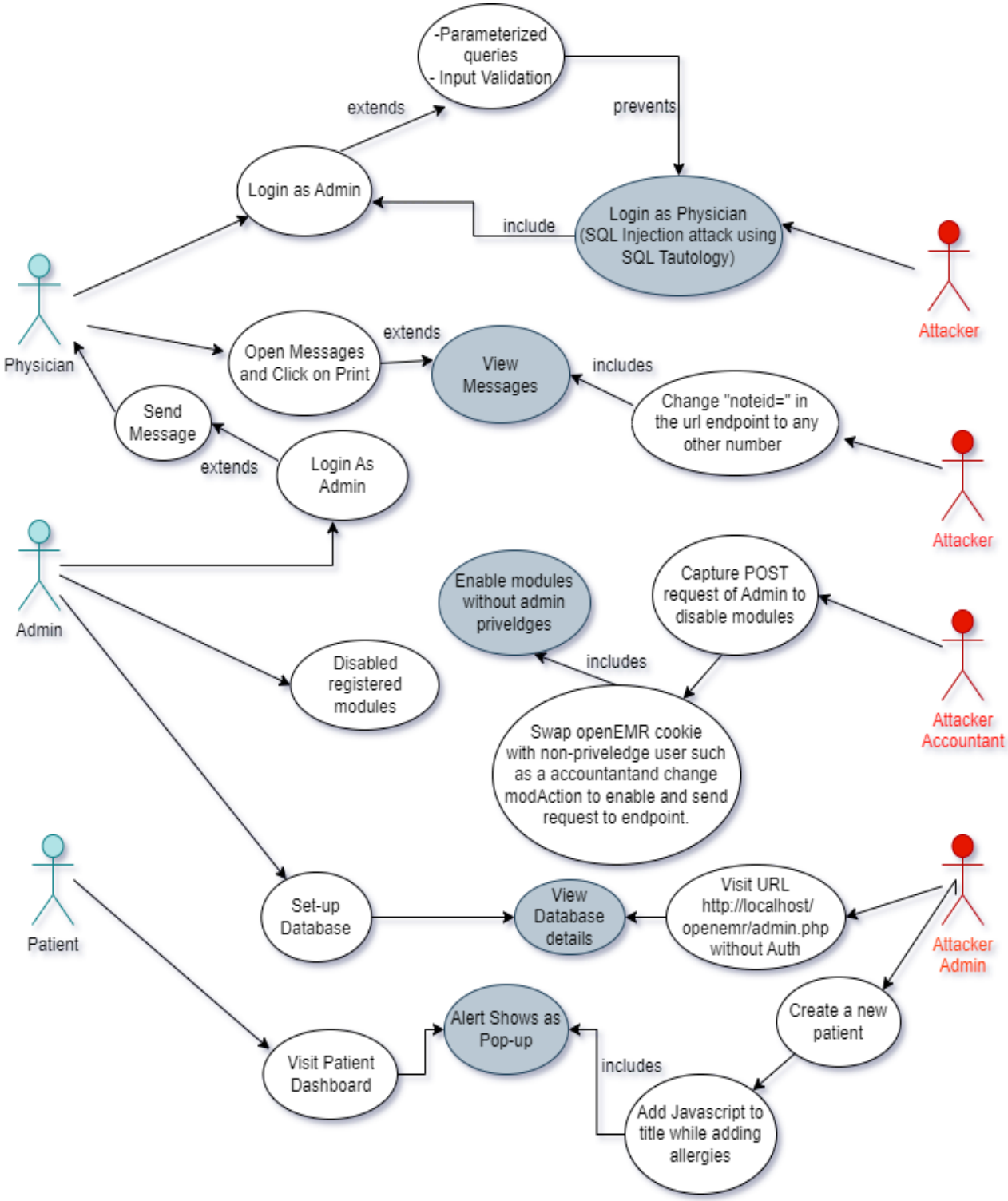
Suggested Change:

- Review the reported issues.
- Fix the issue This could include changing the code logic, changing the code structure, or implementing a particular code fix
- Test the changes made.

Reference:

```
41 =>
PHPStan\Analyser\Error::__set_state(array(
   'message' => 'Access to constant CLAIM_STATUS_CODES_CLP02 on an unknown class OpenEMR\\Billing\\BillingUtilities.
   'file' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\sl_eob_process.php',
   'line' => 275,
   'canBeIgnored' => true,
   'filePath' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\sl_eob_process.php',
   'traitFilePath' => NULL,
   'tip' => 'Learn more at https://phpstan.org/user-guide/discovering-symbols',
   'nodeLine' => 275,
   'nodeType' => 'PhpParser\\Node\\Expr\\ClassConstFetch',
   'identifier' => NULL,
   'metadata' =>
  array (
  ),
)),
42 =>
PHPStan\Analyser\Error::__set_state(array(
   'message' => 'Call to static method updateClaim() on an unknown class OpenEMR\\Billing\\BillingUtilities.',
   'file' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\sl_eob_process.php',
   'line' => 310,
   'canBeIgnored' => true,
   'filePath' => 'C:\\Users\\akrut\\Downloads\\openemr-6.0.0\\openemr\\interface\\billing\\sl_eob_process.php',
   'traitFilePath' => NULL,
   'tip' => 'Learn more at https://phpstan.org/user-guide/discovering-symbols',
   'nodeLine' => 310,
   'nodeType' => 'PhpParser\\Node\\Expr\\StaticCall',
   'identifier' => NULL,
   'metadata' =>
  array (
```

III. Abuse/Misuse Diagram



Template Abuse-Misuse:

Name: OpenEMR Module Disabling Attack
Summary: An attacker intercepts an administrator's POST request to disable or enable OpenEMR modules, potentially allowing them to enable/disable crucial functionality and obtain unauthorized access to patient data.
Author: Akruti, Soha, Shivangi
Date: 27 March 2023

Basic path:

1. An attacker intercepts an administrator's POST request to disable or enable OpenEMR modules.
2. The attacker alters the request (swaps openEMR cookie with non-privileged user such as accountant and changes modAction) so that vital modules are disabled or harmful modules are enabled.
3. The changed request is processed by OpenEMR, potentially disabling vital functionality or allowing unwanted access to patient data.

Alternative paths:

1. The attacker intercepts a POST request sent by a non-admin user who has the required permissions to edit OpenEMR modules.
2. An attacker takes advantage of an OpenEMR vulnerability to circumvent access constraints and get administrative capabilities.
3. The attacker uses social engineering techniques to get admin credentials and carry out the attack.

Capture Points:

1. POST request sent by admin to disable or enable OpenEMR modules
2. Data transmitted between the admin and OpenEMR system

Extension Points:

1. POST request modification to disable crucial modules or enable harmful ones
2. Exploiting vulnerabilities to circumvent access controls or obtain administrative powers

Preconditions:

1. The attacker has gained access to the network traffic between the administrator and the OpenEMR system.
2. The attacker possesses the knowledge and abilities required to intercept and manipulate network communications.
3. The attacker can determine the correct POST request submitted by the administrator to stop or enable OpenEMR modules.

Assumptions:

1. The attacker possesses the knowledge and abilities required to intercept and manipulate network communications.

2. The attacker can determine the proper POST request submitted by the administrator to stop or allow OpenEMR modules.
3. The attacker is motivated to carry out the attack in order to get unauthorized access to patient data.

Worst case threat (postcondition):
The attacker has the ability to disable important modules or enable malicious ones, potentially resulting in a data leak or disruption of critical healthcare services.

Capture Guarantee (Postcondition):
The system cannot ensure the capture or detection of this form of attack since it relies on network security safeguards outside of the application.

Related Business Rules:
- Only authorized users with the proper rights can modify OpenEMR modules.
- Changes to OpenEMR modules must be logged and audited.

Potential Misuser Profile:
The attacker is a threat actor, either external or internal, who has access to the network communication between the administrator and the OpenEMR system.

Stakeholders and threats:
- Stakeholders: Healthcare providers, patients, and other users of the OpenEMR system.
- Threats:
    - Unauthorized access to patient data,
    - Disruption of critical healthcare services,
    - Damage to the reputation of the healthcare provider.

Scope: Entire business and business environment
Abstraction Level: Mis-user goal
Precision Level: Focussed