
# Security Testing Report

## EDI\Interoperability Solutions

## XDS 2023-1

# Table of Contents

# Document Version Control

| | | |
|---|---|---|
| **Name of the document : XDS 2023-1 Security Testing Report** | | |
| **Version:** 7.0 | **Intake ID:** | 2648 |
| **Document Definition: This document highlights the vulnerabilities currently existing in the application under scope. It also documents possible actions to be taken to reduce/eliminate the vulnerabilities.** | **Document ID:** | PRHC/C40/SVN/86070 |
| **Author:** Raj Kiran Rudrapati | **Effective Date:** | 09/Jun/2023 |
| **Reviewed by:** Aravind C Ajayan | | |

# Document History

| Version | Date | Author | Section | Changes |
|---|---|---|---|---|
| 0.1 | 20/Jul/2020 | Narendra Makkena | Complete | Initial Draft |
| 1.0 | 20/Jul/2020 | Taksh Medhavi | Complete | Addition & review |
| 1.1 | 01/Mar/2021 | Shabana Bagum | Complete | Initial Draft |
| 1.2 | 02/Mar/2021 | K K Ashwin | Complete | Addition & review |
| 2.0 | 03/Mar/2021 | Narendra Makkena | Complete | Final review |
| 2.1 | 03/March/2022 | Shibija K | Complete | Initial Draft |
| 2.2 | 04/March/2022 | K K Ashwin | Complete | Addition & review |

| | | | | |
|---|---|---|---|---|
| 3.0 | 04/March/2022 | Pranati Mohanty | Complete | Final review |
| 3.1 | 27/May/2022 | Sreerag M | Vuln 8.2,8.4,8.6 and 8.11 | Retest |
| 3.2 | 27/May/2022 | Shibija K | Vuln 8.2,8.4,8.6 and 8.11 | Addition & review |
| 4.0 | 27/May/2022 | Pranati Mohanty | Vuln 8.2,8.4,8.6 and 8.11 | Final Review |
| 4.1 | 27/Aug/2022 | Kartik Lalan | Complete | Addition & review |
| 5.0 | 29/Aug/2022 | Shibija K | Complete | Final Review |
| 5.1 | 31/Jan/2023 | Ashwin K K | Vuln 8.2,8.4,8.6 and 8.11 | New feaure and retest |
| 5.2 | 01/Feb/2023 | Shibija K | Complete | Addition & technical review |
| 6.0 | 01/ Feb /2023 | Pranati Mohanty | Complete | Final Review |
| 6.1 | 08/Jun/2023 | Raj Kiran Rudrapati | Complete | Issue revalidation & Rapid test |
| 7.0 | 09/Jun/2023 | Aravind C Ajayan | Complete | Addition & Review |

# Distribution List

| User/Department/Stakeholder | E-Mail ID |
|---|---|
| **Project Owner and PSO** | subhash.naga@philips.com;<br>Stephanie.Heidstra@philips.com;<br>moin.creemers@philips.com;<br>ManishKumar.MachingalSukumar@philips.com;<br>Stephanie.Heidstra@philips.com;<br>chidamber.kumar_1@philips.com;<br>abhishek.kumar.pathak@philips.com;<br>revathi.r@philips.com |

# 1. Definitions & Abbreviations

| Term | Explanation |
|---|---|
| SCoE | Security Center of Excellence |
| TLS | Transport Layer Security |
| SSL | Secure Socket Layer |
| XSS | Cross Site Scripting |

The severity of every vulnerability has been calculated by using industry standard **Common Vulnerability Scoring System (CVSS)** used for assessing the severity of computer system vulnerabilities. CVSS provides a way to capture the principal characteristics of a vulnerability and produce a numerical score (Scores range from 0 to 10, with 10 being the most severe) reflecting its severity. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organization properly assess and prioritize their vulnerability management processes.

The severity rating for the numerical values are mapped below:

| | |
|---|---|
| None | 0.0 |
| Low | 0.1 - 3.9 |
| Medium | 4.0 - 6.9 |
| High | 7.0 - 8.9 |
| Critical | 9.0 - 10.0 |

The **Severity** and **CVSS vector** of each vulnerability is calculated using the CVSS V3 **Base Score Metrics** Calculator located here. Vulnerabilities identified during security assessment are classified into standardized categories. Refer following table for more information:

Categories for vulnerability classification

| | |
|---|---|
| Web application security assessment | OWASP Top Ten - 2021 |
| Mobile application security assessment | OWASP Top Ten - 2016 |
| IoT/Hardware security assessment | OWASP Top Ten – 2014 |

# 2. System Details & Architecture

Brief about the product architecture:

Philips Interoperability Solutions (formerly Forcare) is an open-standard-based interoperability software solutions for fast and flawless data flows between medical systems and information sources at the departmental and enterprise levels, as well as Health Information Exchanges (HIEs) across health systems.

The tested components: ForView, ForAudit ForImageUpload and ForAdmin web application.

**ForAdmin** - ForAdmin is a web-based application for system administrators. This can be used for configuring and monitoring the Forcare application components in your network. With ForAdmin, you can make configuration changes at runtime without requiring restarting your server. Component configurations are stored in XML files that are validated to prevent basic configuration errors.

**ForView-** ForView is an application for care providers that lets them find, view, and share patient-oriented clinical information. ForView complements existing clinical information systems, adding the ability to view documents and images from a variety of sources that are not limited to one specific institution or document type.

**ForAudit-** ForAudit is an IHE ATNA Audit Repository. It captures audit logs sent via UDP or TCP/SSL (syslog, per IHE ITI ATNA) and supports HTTP or HTTPS based audit transactions.

**ForImageUpload-** ForImageUpload is an application to upload Dicom/Zip files of images.

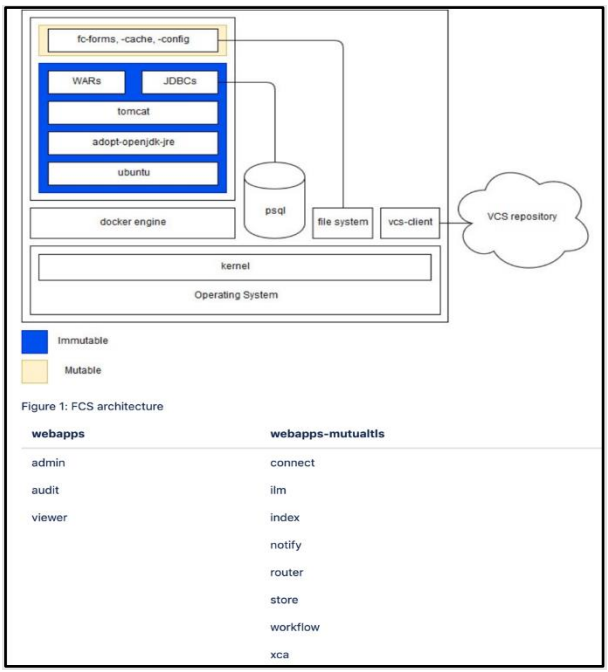



Figure 1: FCS architecture

| webapps | webapps-mutualtls |
|---|---|
| admin | connect |
| audit | ilm |
| viewer | index |
| | notify |
| | router |
| | store |
| | workflow |
| | xca |

The environment provided for security testing is a Pentest Environment.

# 3. Scope

The scope of this security assessment is to perform **Grey box** security testing to find security threats that may come from a malicious outsider or insider user of the **XDS 2023-1.** Security testing on **Web applications (Viewer,Audit,Admin&ImageUpload)** of **XDS 2023-1** is performed.

The following list includes some examples of major activities performed during the assessment:

**Web Application:**

- Crawl through complete scope of the web application/service space and identify for any unauthenticated URL or directory.
- Check for all input injection-based attacks across all the possible entry fields in Web API.
- Exploiting any known component vulnerability or service misconfiguration.
- Reviewing the transport layer security implemented.

*Follow "*Test case execution*" section to get the detailed about test*

| Type | Scope of Assessment | | |
|---|---|---|---|
| Web Application | XDS | URL | https://server_url/admin<br>https://server_url/audit<br>https://server_url/imageupload<br>https://server_url/viewer |
| | | Version | 2023-1 |
| | | Environment | Pentest |
| | | User Roles | root (self creatable user roles) |

* server_url – During the assessment, the server IP was changed couple of times. The IP's are listed below. So in this document, url is referred to as 'server_url'.

IP's:

https://3.67.186.182/

https://18.196.27.233/

https://3.71.200.223/

**Note (09/06/2023):** The scope of the testing is to evaluate 2 new features deployed as part 2022-4 release and revalidate previous open vulnerabilities and a rapid testing of the application functionalities in limited time frame & below changes:

- The scope of the testing is to validate the application functionalities related to the issues (DEF-6166, DEF-6242, DEF-6284, DEF-6709, DEF-6813, DEF-6816, DEF-6831) and a rapid testing of the application functionalities.
- Scope of the testing is to make sure that the fixes do not introduce any security vulnerabilities.

# 4. Out of Scope

Below mentioned items are out of scope for the current security assessment:

- Source code review
- Cloud Testing
- Docker Testing
- Complete security assessment of 4 portals

**Note:** We have covered the testing of **XDS 2023-1** in the environment provided and the results are valid if the same environment is replicated. Re-run the tests if a new propagation of the environment is made.

# 5. Executive Summary

Security Center of Excellence (SCoE) team is engaged in activities to conduct security assessment of **XDS 2023-1** which included **Web applications** in scope. The purpose of the engagement is to evaluate the security of the **XDS 2023-1** against industry best practice criteria.

Note: Only highlights of important vulnerabilities are described below. Please refer 'Vulnerability Summary' section for complete detailed list of vulnerabilities.

During the security assessment following factors are found with consideration for significant improvement:

- Using known vulnerable components
- Weak input validations
- Likelihood of cross site scripting
- Weak cipher suite used
- Multiple security misconfigurations
- Weak password policy on Foradmin portal

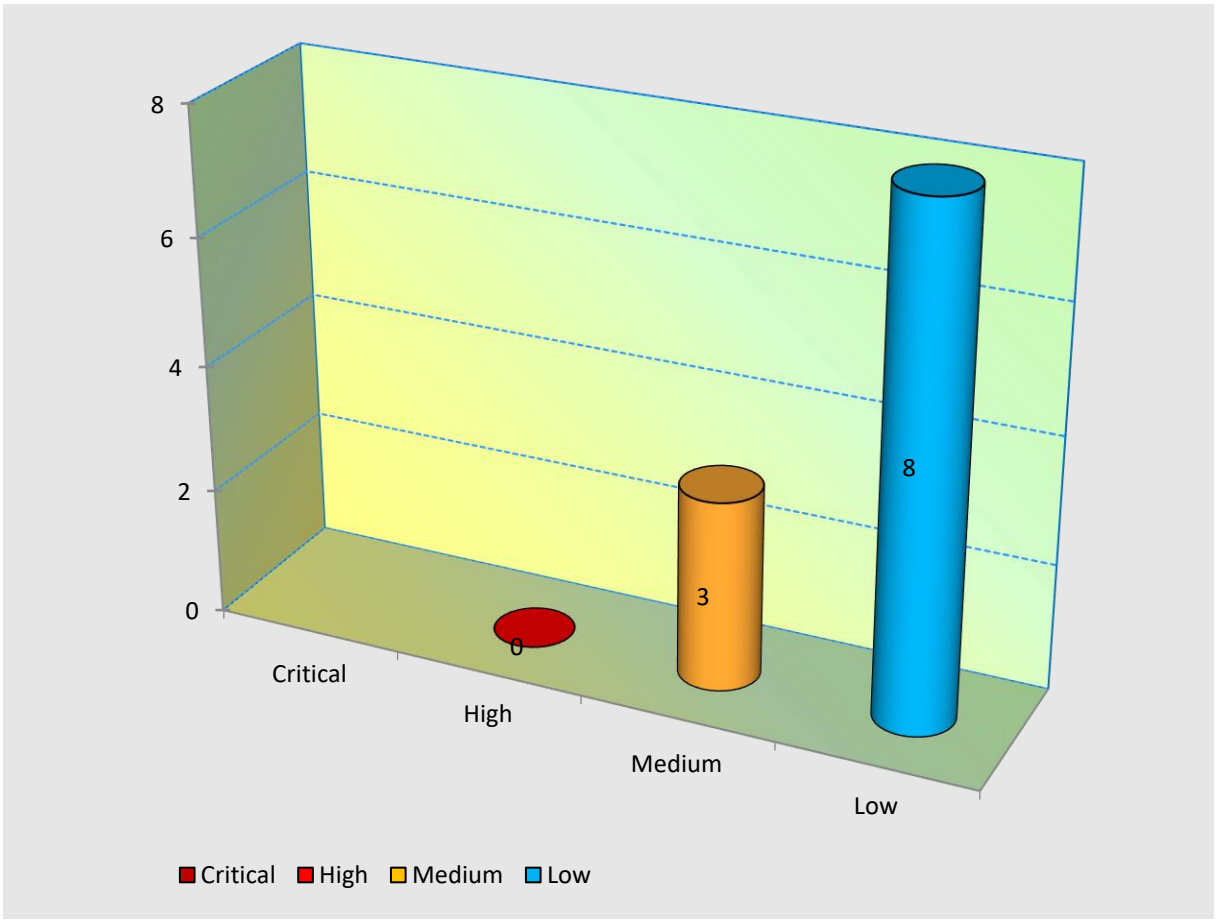
During the security assessment of the product, security issues in the below areas are not found:

- Cross site request forgery
- Token Management

## VULNERABILITY SUMMARY CHART

The graph below shows a summary of the number of vulnerabilities and their severities.

**Note:** The vulnerabilities mentioned in this report are technical vulnerabilities only. The Product Security Risk Assessment would report the business risks associated with these vulnerabilities.



# 6. Vulnerability Summary

The findings and vulnerabilities from the assessment are explained in the below table:

| Finding No. | Vulnerability Title | Technical Risk | Impacted Area | CVE ID* | Status | Status(31st Jan 23) | Status(6th Jun 23) |
|---|---|---|---|---|---|---|---|
| 30132 | Using Known vulnerable components | Medium | Webapp | Refer 8.1 | Open | Open | Open |
| 79143 | Broken Access Control | Medium | Webapp | NA | Open | Open | Open |
| 30136 | Weak Password Policy | Medium | Webapp | NA | Open | Open | Open |
| 77505 | DOM Cross Site Scripting | Low | Webapp | NA | Open | Open | Open |
| 30145 | Weak Input Validation | Low | Webapp | NA | Open | Open | Open |
| 30209 | Improper error & exception handling | Low | Webapp | NA | Open | Open | Open |
| 37688 | Sensitive information in the URL | Low | Webapp | NA | Open | Open | Open |
| 30128 | HTTP Strict Transport Security (HSTS) Not Implemented | Low | Webapp | NA | Open | Open | Open |
| 30146 | No Account Lockout Policy | Low | Webapp | NA | Open | Open | Open |
| 30125 | Reflected Cross-Site Scripting (XSS) | Low | Webapp | NA | Open | Open | Open |
| 83850 | Weak SSL/TLS Configuration | Low | Webapp | NA | - | New | Open |

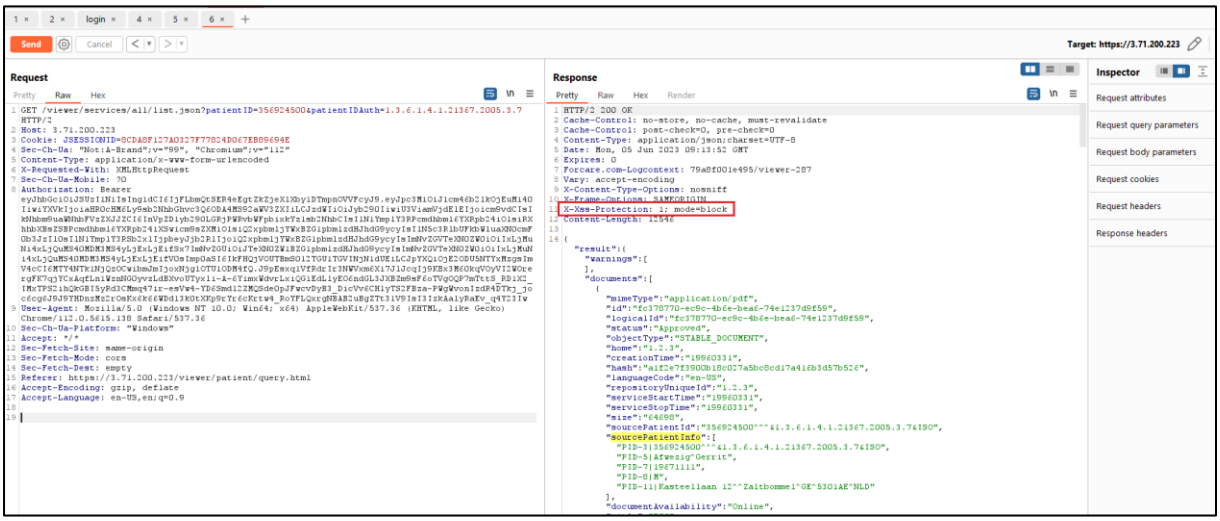
*CVE ID are mentioned for the vulnerabilities which has a known external CVE.

# 7. Observations

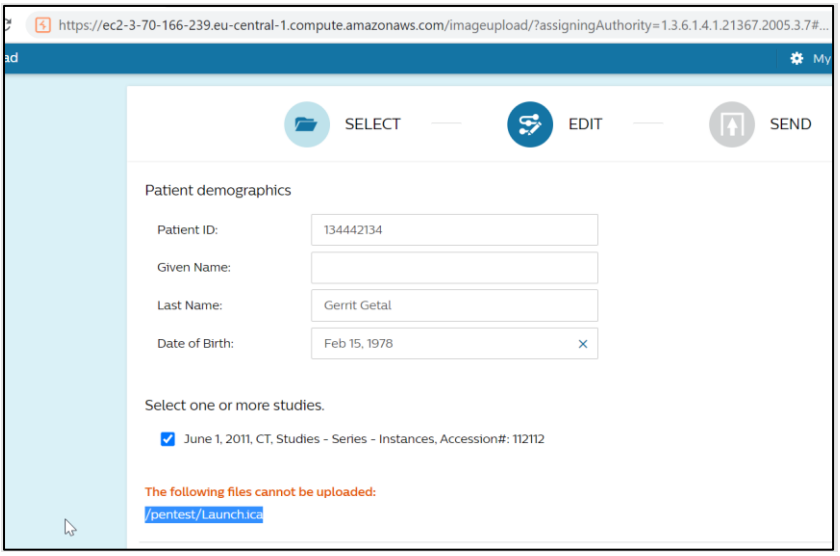
Below mentioned observations are not considered as vulnerability but informative to the business.

*Observations which shows good implementation or best practice identified:*

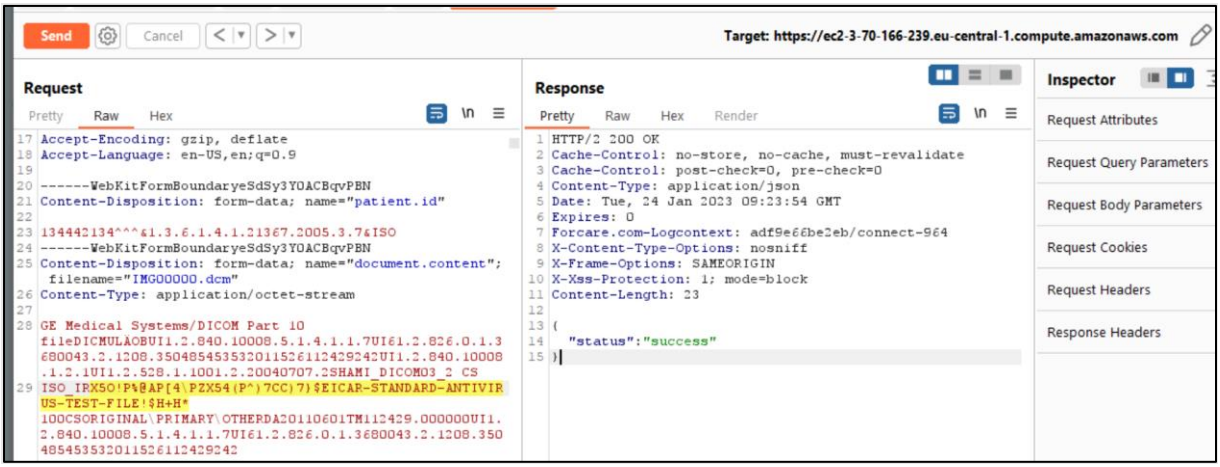
- XSS protection flag is enabled.




- Image Upload feature has proper restriction in place for rejecting unintended malicious files as part ZIP and DICOMDIR functionality.

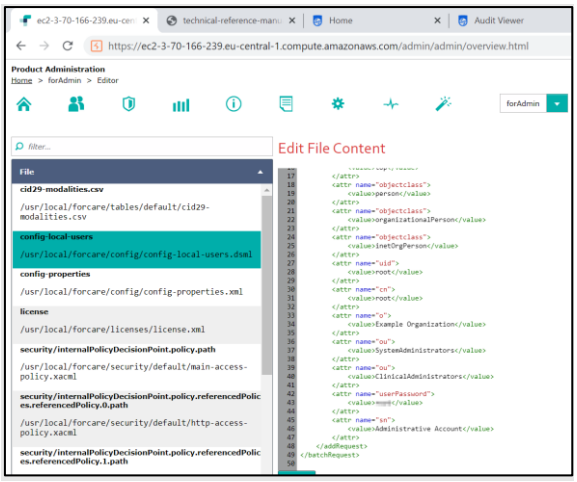



- File upload accepts EICAR virus characters but the server quarantines this particular portion of file. When we download this upload again, one file where we injected the payload is not present in the ZIP file.

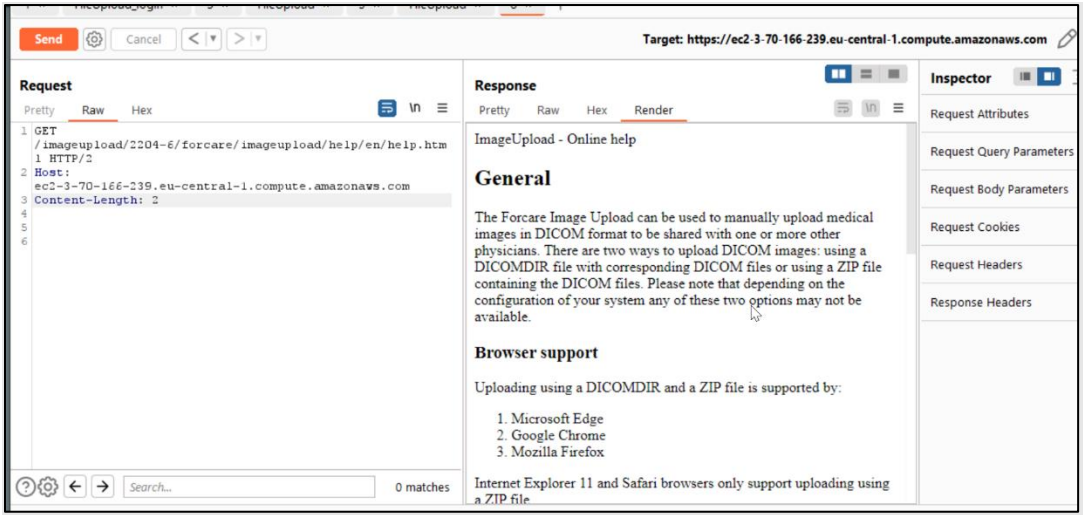



*Observations which shows missing best practice or possible weak implementation (this may/may not be direct active threat):*

- No email verification while registering the user.
- Password is created by admin user, No password change option available at other applications.
- Application is allowing simultaneous session logons.
- Even though, there are 2 sets of authentication parameters (JSESSIONID & Authorization Token), some of the functionalities of the application is accessible with just either one of the parameters.
- Also, we highly recommend enforcing jwt token for authentication/authorization rather than relying up on either of the above-mentioned authentication mechanisms.
- Password is in plain text at admin portal in the configurations.



- ImageUpload Online help page does not require any authentication. Since there is no PII or PHI we are marking this as Observation.




RabbitMQ

- Default credentials found in environment variable for rabbitmq

- The bind mounts has Read Write enabled which is not a good implementation from security standpoint.


# 8. Detailed Vulnerability Report

## 8.1 Webapp: Using Known Vulnerable Components

| | |
|---|---|
| **Vulnerability Title** | Using Known Vulnerable Components |
| **Vulnerability Category** | A6 Vulnerable and Outdated Components |
| **Severity** | **Medium** |
| **CVSS V3 Calculation** | CVSS Base Score: 4.8<br>CVSS:3.0/ AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N |
| **Description** | **Vulnerability Description:** During the security assessment of the product, it is observed that the web application is using compontents with known vulnerabilities and associated CVEs.<br><br>Technologies are like jquery-1.2.6, dojo 1.9.6 and AngularJS v1.8.2 have known CVE IDs:<br><br>jquery-1.2.6, AngularJS v1.8.2, and dojo 1.9.6<br><br>• moment.js 2.29.3 - CVE-2022-31129<br>• jquery-1.2.6 - https://nvd.nist.gov/vuln/detail/CVE-2011-4969<br>• AngularJS v1.8.2 - angular 1.8.2 vulnerabilities \| Snyk<br>• Dojo 1.9.6 - https://security.snyk.io/package/npm/dojo/1.9.2<br><br>AngularJS LTS discontinued.<br><br>Discontinued Long Term Support for AngularJS \| by Mark Thompson (@marktechson) \| Angular Blog<br><br>**Retest (02-Jun-2023):**<br>It is observed that the server makes use of the mentioned vulnerable component.<br>jquery-1.2.6, AngularJS v1.8.2, moment.js 2.29.3 and dojo 1.9.6 are being used.<br>jQuery 1.x and 2.x are End-of-Life and no longer receiving security updates<br>**Retest (30-Jan-2023)**:<br>It is identified that the server makes use of AngularJS which has reached end of life.<br>Server banner disclosure is also happening as part of server response. It is observed Apache Tomcat/9.0.68 is been used by the application. |

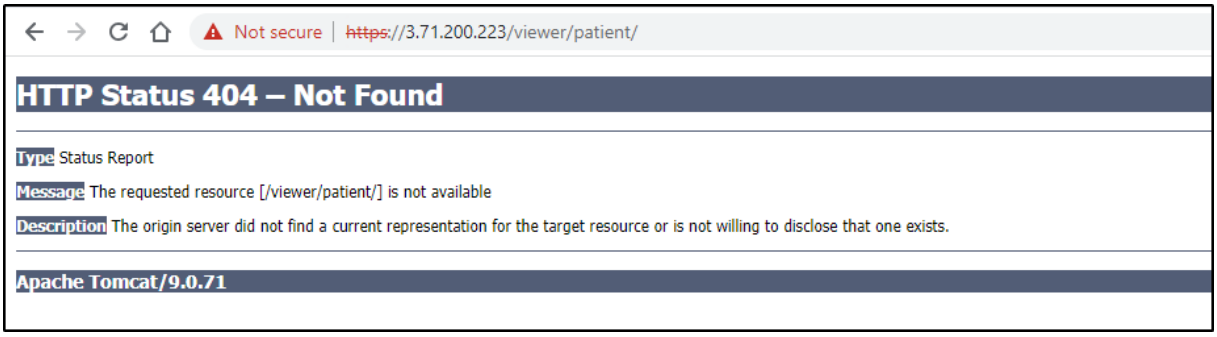
| | |
|---|---|
| | **Exploitability Rational:** Published vulnerabilities have a greater likelihood of exploitation by attackers due to readily available proof-of-concept code that exploits the issue, or integrates the exploits into freely available testing tools.<br><br>**Impact Rational:** Depending on the nature of known vulnerabilities, this can allow an attacker to compromise the server and any data stored within. |
| **Affected Systems/IP Address/URL** | **Retest (02-Jun-2023):**<br>https://server_url/admin/<br><br>https://server_url/viewer/<br><br>https://server_url/audit/<br><br>**Old POC:**<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/2204-47/js/angular/angular.min.js |
| **Recommendation** | Service version is vulnerable to an attack, take preventative measures to mitigate the vulnerability until an upgrade or patch is released. |
| **Status** | **Open** |

**Steps to Reproduce**

Steps same as before


audit_login_pg_only.html

viewer_retirejs.html

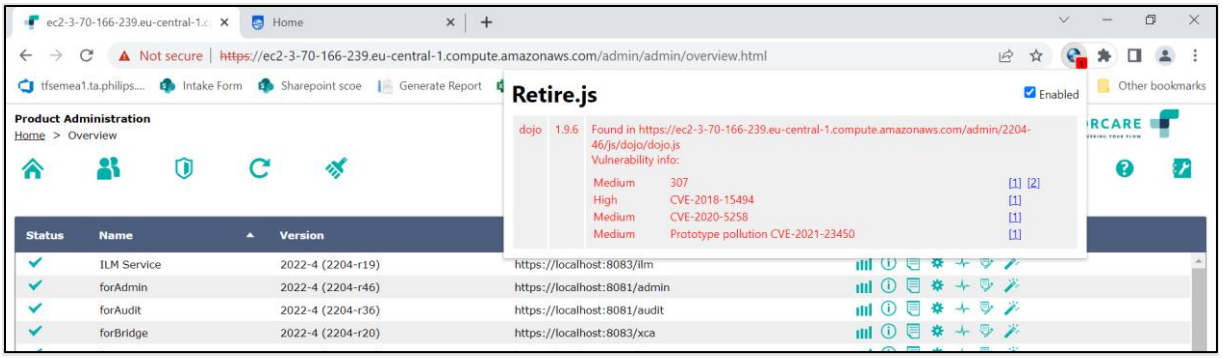



Old POC:

- Launch the application and run retire.js plugin.
- Login to any application ForAdmin, ForView.

ForView Portal


ForAdmin portal




Banner grabbing of Apache Tomcat server version.

## 8.2 Webapp: Broken Access Control

| | |
|---|---|
| **Vulnerability Title** | Broken Access Control |
| **Vulnerability Category** | A1 Broken Access Control |
| **Severity** | **Medium** |
| **CVSS V3 Calculation** | CVSS Base Score: 6.5<br>CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N |
| **Description** | **Vulnerability Description:**<br><br>The application allows lower privileged users to access resources or perform actions which is available to a higher level account. This usually occurs when the server does not perform authorization/entitlement checks on each request to ensure that the user has the appropriate privileges before executing the request, or when those checks are made based on values that can be tampered with on the client-side.<br><br>It is observed that the user without SYSTEM_PHI_READ role defind can view/read the following API endpoints which consists of sensitive information like, backend components/versions, encryption password etc.<br><br>**Retest (02-Jun-2023):**<br>It is observed that the issue is still persistent.<br>During the assessment, it is observed that after some time, the token becomes unauthorized but still able to retrieve data.<br><br>**Retest (31-Jan-2023)**:<br>It is observed that user with SYSTEM_PHI_READ has view option on backend configuration which includes sensitive information. Issue still persists.<br><br>**Exploitability Rational:**<br><br>The attacker can be any user who has access to the forcare admin application.<br><br>**Impact Rational:**<br><br>Depending on the nature of the information, a malicious user may obtain personally identifiable information (PII), private user data or information which can allow user impersonation (in the event of credential or session identifier). |

| | |
|---|---|
| **Affected Systems/IP Address/URL** | Retest (02-Jun-2023):<br><br>https://server_url/admin/services/admin/components/viewer/status/retrieve.json?format=true<br><br>https://server_url/admin/services/admin/components/admin/config/properties.json<br><br>https://server_url/admin/services/admin/components/audit/editor/list.json<br><br>https://server_url/admin/services/admin/components/viewer/status/retrieve.json?format=true<br><br>https://server_url/admin/services/admin/components/admin/config/properties.json<br><br>https://server_url/admin/services/admin/queue/retrieve.json<br><br>https://server_url/admin/services/admin/components/audit/editor/list.json<br><br>https://server_url/viewer/2301-14/js/forcare/bo/PatientId.js<br><br>Old POC<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin/services/admin/components/viewer/status/retrieve.json?format=true<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin/services/admin/components/admin/config/properties.json<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin/services/admin/components/audit/editor/list.json<br><br>Note: This is an application wide issue. Instances are not limited to the above items. Fix should be applied across the platform. |
| **Recommendation** | Check and verify the privilegies of the user and check or verify before serving the response. |
| **Status** | **Open** |

**Steps to Reproduce**

**Retest (02-Jun-2023):**

**Case1:**

The steps same as before

**Case2:**

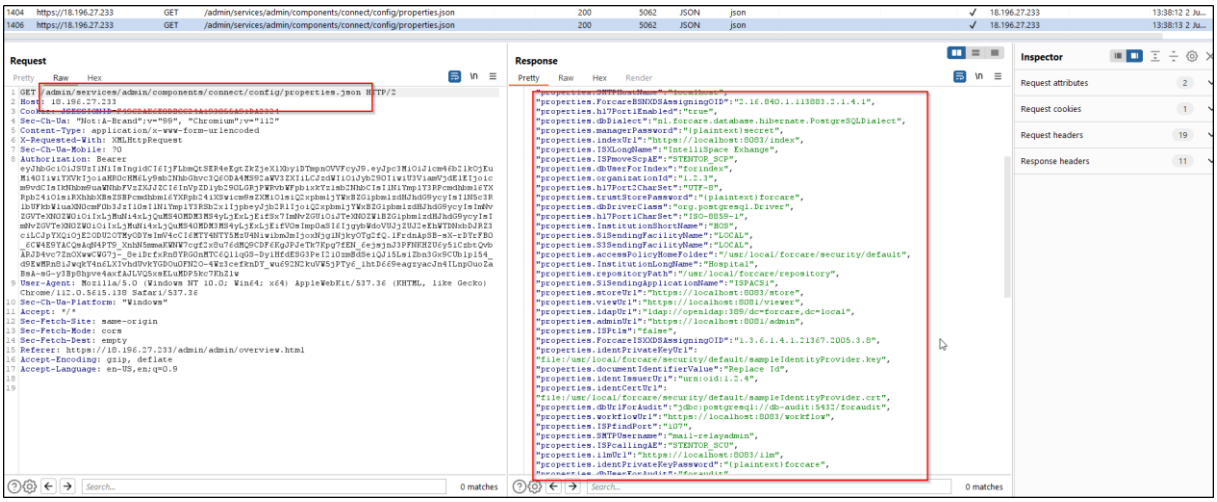
Step1: Configure the browser with a proxy tool like Burp Suite and log into the foradmin application.

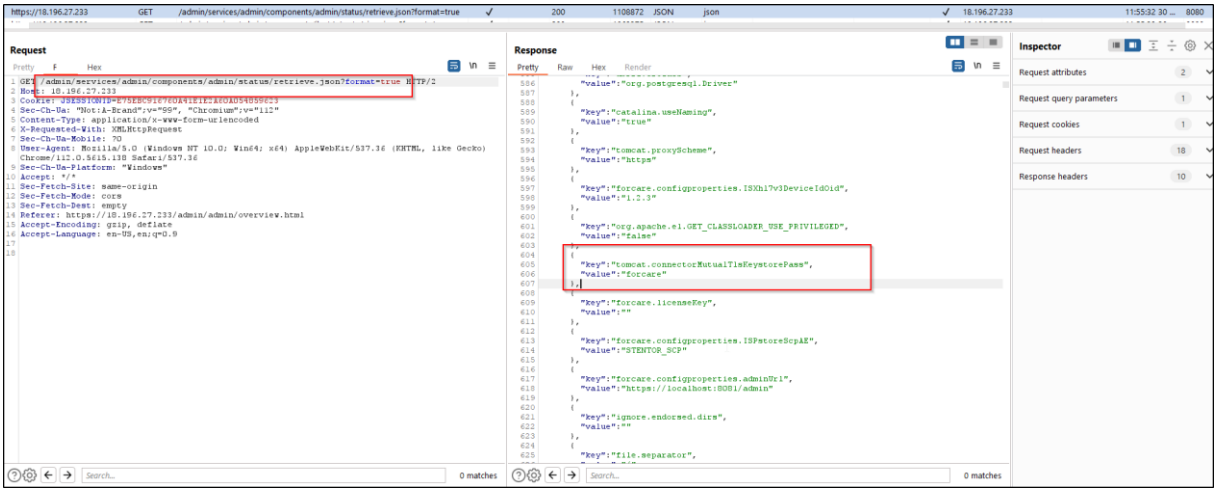
Step2: Leave the setup idle for about 5-10mins, the token becomes unauthorized.

Step3: Send the retrieve data request to repeater and observe the behavior.

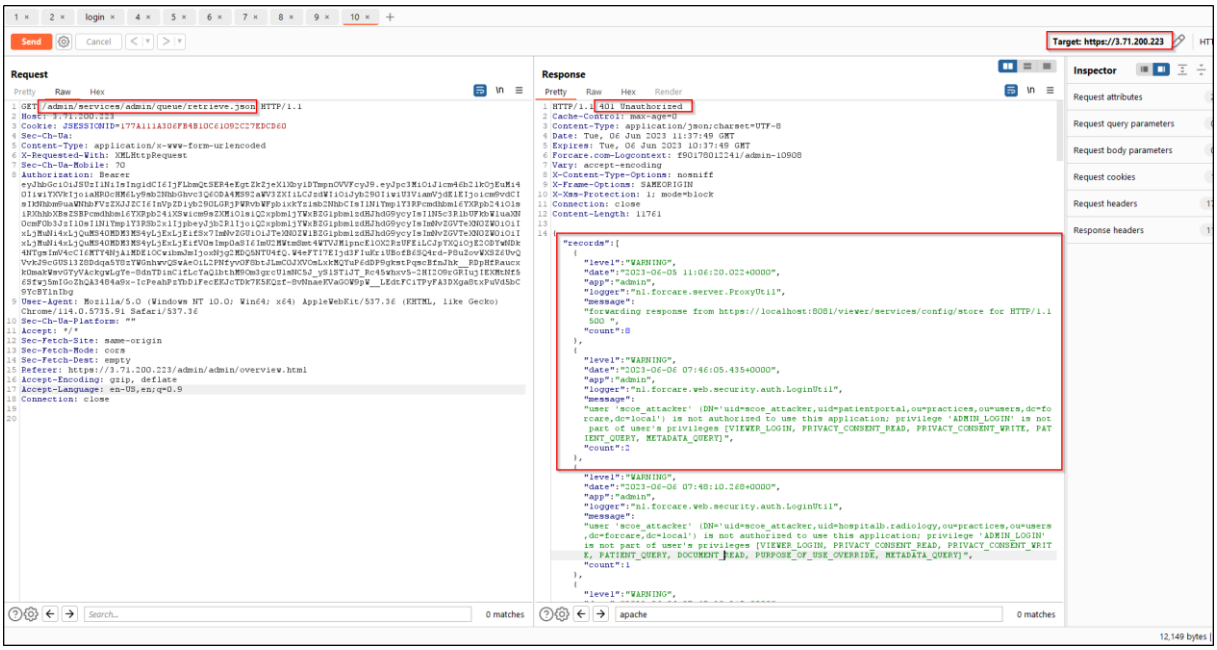
**Supportive Evidence:**

**Case1:**





**Case2:**

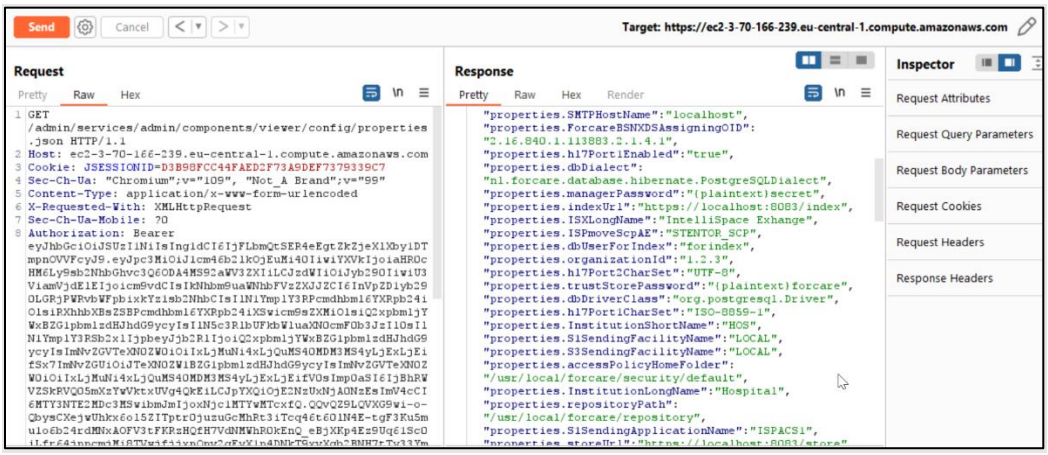


**Old POC:**

1. Configure your browser to use a proxy tool such as Burp.
2. Log in to the forcare admin application as user without SYSTEM_PHI_READ.
3. It is observed that the user without SYSTEM_PHI_READ role defined can view/read the following api endpoints which consists of sensitive information like, backend components/versions, encryption password etc.

**<u>Supportive Evidence:</u>**

A user with SYSTEM_PHI_READ has view option on backend configuration which includes sensitive information.




## 8.3 Webapp: Weak Password Policy

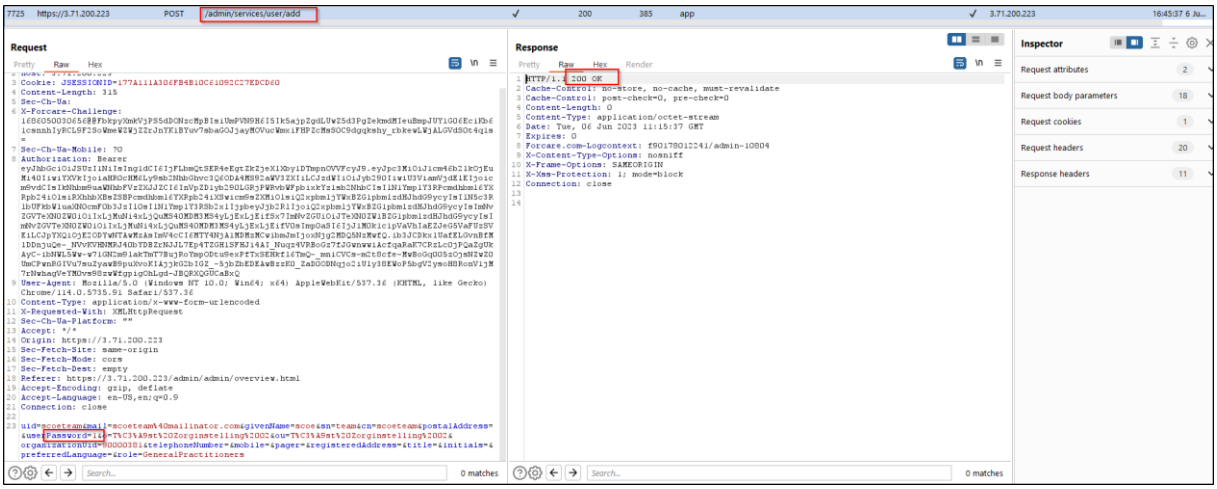
| | |
|---|---|
| **Vulnerability Title** | Weak Password Policy |
| **Vulnerability Category** | A6 - Security Misconfiguration |
| **Severity** | **Medium** |
| **CVSS V3 Calculation** | CVSS Base Score: 4.6<br>CVSS:3.0/AV:A/AC:H/PR:H/UI:R/S:U/C:L/I:H/A:N |
| **Description** | **Vulnerability Description:**<br><br>The application does not enforce a strong password policy to prevent malicious users from manually guessing or brute-forcing legitimate account passwords. Weak password policies include those that allow passwords consisting of common dictionary words, commonly-used passwords (For example., 1234), passwords that contain the associated username, sequential characters, and passwords shorter than 9 characters. By allowing users to create easily-guessable passwords, an attacker with minimal knowledge of registered users and username formats could crack passwords through the use of any of several techniques. In an online attack, an attacker can use consecutive login attempts to determine simple passwords. In an offline attack (For eaxmple., if the attacker has gained access to the raw contents of the password database through some other means), the attacker can employ richer techniques such as pre-computed hash attacks, free of rate-limiting and account-locking protections that might be employed against online password brute-forcing attacks.<br><br>**Retest (05-Jun-2023):** It is observed that the issue persists.<br><br>**Retest (31-Jan-2023):** It is observed that just like previous time there is no password policy in place, even password of single character is accepted.<br><br>**Exploitability rational**<br><br>Weak passwords may be easily guessed. This increases the likelihood a user's account may be compromised by an attacker. Once compromised, an attacker can have full access to the victim's account, potentially including the ability to modify settings, features and passwords. If the target account holds |

| | |
|---|---|
| | administrative privileges, the attacker can modify data for other users and/or the entire system<br><br>**Impact rational:** Credentials can be easily brute force. . Due to the prevalence of password reuse, a compromised password may also provide an attacker with credentials that can be used to attack other systems the victim uses the same credentials to access. |
| **Affected Systems/IP Address/URL** | Retest (05-Jun-2023):<br><br>https://server_url/admin/<br><br>https://server_url/audit/<br><br>https://server_url/viewer/<br><br>Old:<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin/<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/audit/<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/ |
| **Recommendation** | **Recommendation:**<br><br>Implement a strong password complexity policy. A strong password policy is one which combines rules to prevent easily-guessable passwords from being used while also ensuring that passwords contain sufficient entropy. A password policy which provides a large set of restrictions can ultimately result in a smaller potential pool of passwords, lowering the amount of time necessary to guess a password through brute-force attacks. Conversely, an overly permissive policy allows users to create easily-guessable passwords. Put constraints in place to prevent users from choosing easily-guessable passwords at the time of creation, specifically those that are targeted by well-known dictionary attacks. This can minimize the likelihood that an attack may be successful if an attacker attempts to guess commonly-used passwords or employs an automated dictionary attack against a particular user. |

| | |
|---|---|
| | In the event that company policy does not stipulate password requirements, or the existing requirements are weak, consider employing the following password complexity requirements:<br>• Passwords must be at least nine (9) characters long.<br>• Passwords must contain some combination of at least three (3) of the following classes of characters: lowercase, uppercase, numeric, and “special” (For example., !, @, #, $, %, ^, etc.) characters.<br>• Passwords should not be a dictionary word in any language, slang, dialect, jargon, etc.<br>• Passwords should not be based on personal information, etc. |
| **Status** | **Open** |

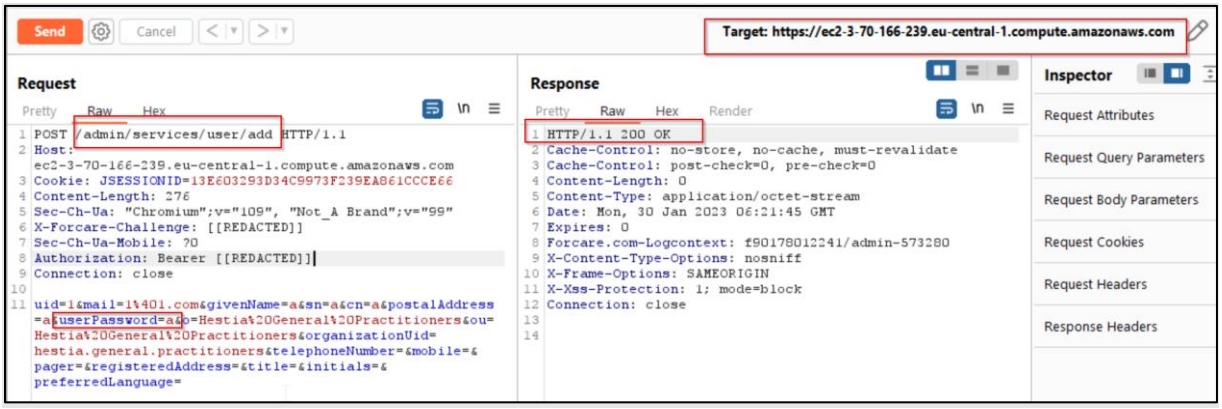
**Steps to Reproduce**

**Retest (05-Jun-2023):**

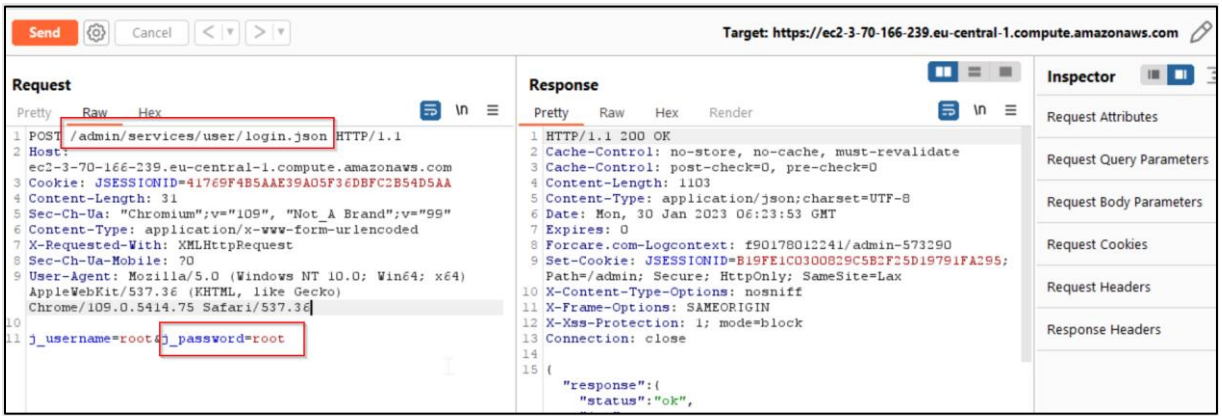



**Old POC:**

Weak password accepted for add user feature in ForAdmin portal.




Weak login password for the portals.

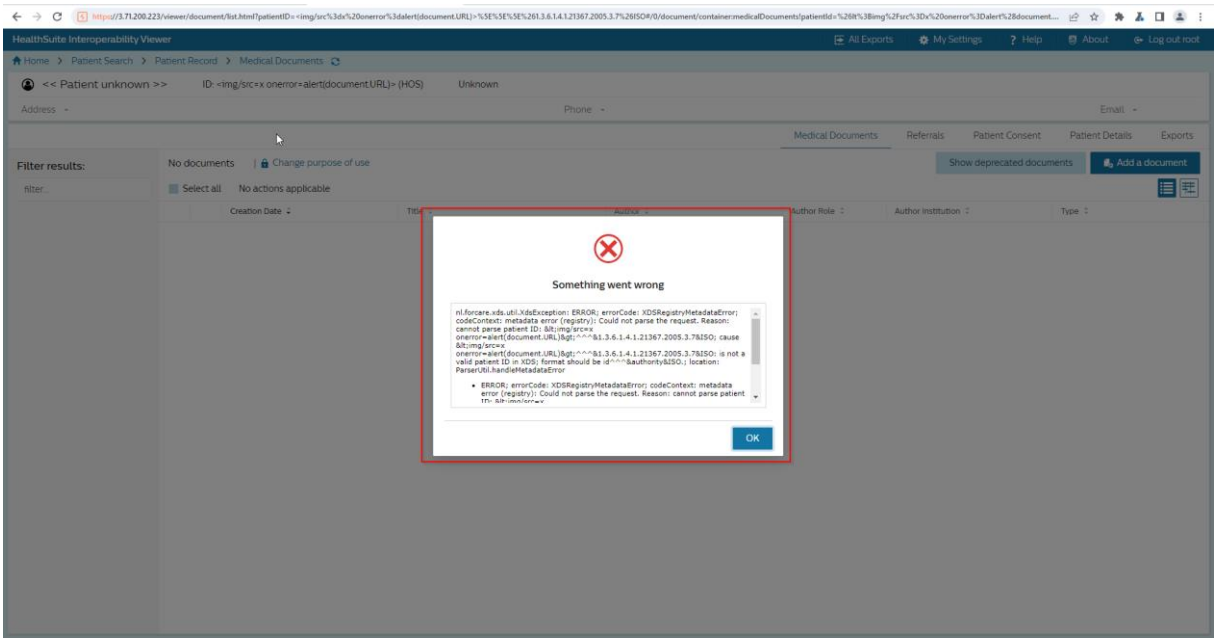


## 8.4 Webapp: DOM Cross-Site Scripting (XSS)

| | |
|---|---|
| **Vulnerability Title** | DOM Cross-Site Scripting (XSS) |
| **Vulnerability Category** | A3- Injection |
| **Severity** | Low |
| **CVSS V3 Calculation** | CVSS Base Score: 3.5<br>CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N |
| **Description** | **Vulnerability Description:** During the security assessment it is observed that the application is vulnerable to DOM XSS attack.<br><br>DOM based XSS is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner. That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.<br><br>**Retest (03-Jun-2023)**: During the retest, it is observed that the issue persists.<br><br>Reducing the severity to low as XSS is no more happening.<br><br>**Retest (31-Jan-2023)**: It is identified that DOM based XSS payload uncaught exception error indicating fix is not properly implemented yet.<br><br>**Exploitability Rational:** For a successful exploitation of the issue, the victim should not be logged in to the application.<br><br>**Impact Rational:** DOM Cross-Site Scripting vulnerabilities give the attacker control of HTML and JavaScript running the user's browser. The attack can alter page content with malicious HTML or JavaScript code. The attacker can arbitrarily alter page content displayed to the victim and can execute application functions using the victim's application identity if the victim is authenticated to the application. |

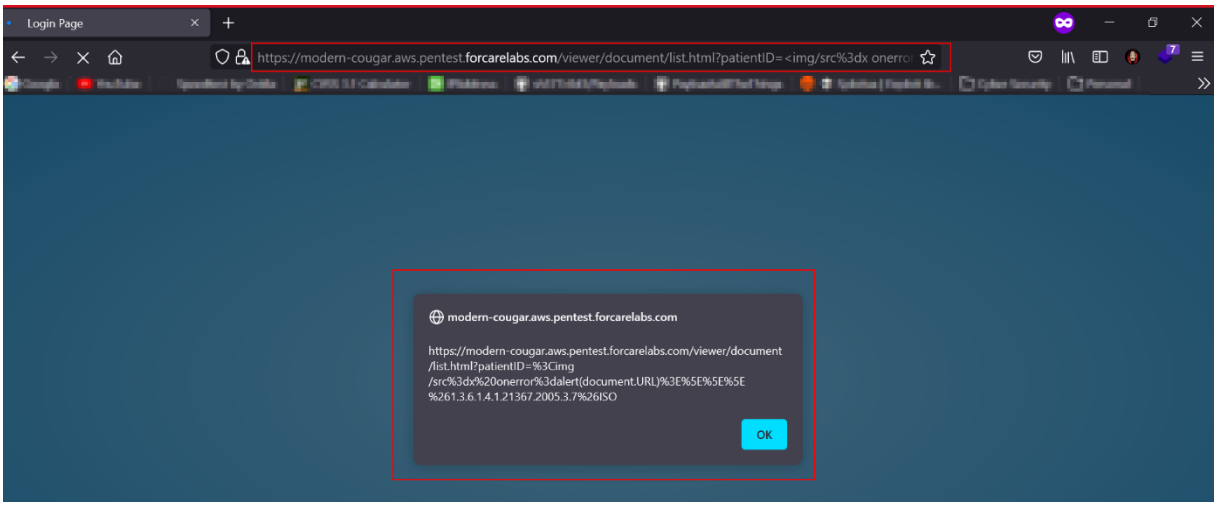
| | |
|---|---|
| **Affected Systems/IP Address/URL** | Retest (03-Jun-2023):<br><br>https://server_url/viewer/document/list.html?patientID=%3Cimg/src%3dx%20onerror%3dalert(document.URL)%3E%5E%5E%5E%261.3.6.1.4.1.21367.2005.3.7%26ISO<br><br>Old:<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/document/list.html?patientID=%3Cimg/src%3dx%20onerror%3dalert(document.URL)%3E%5E%5E%5E%261.3.6.1.4.1.21367.2005.3.7%26ISO |
| **Recommendation** | It is recommended to have proper input validation, also checking syntax-semantic (For example, a name field allowing special chars, number field accepting anything other than numeric digits., or allowing custom URL schema.<br><br>Validation at Server side is required, you can bypass client side within javascript.<br><br>Encode the output with Encoder.encodeForHTML and Encoder.encodeForJS before making dynamic updates to HTML.<br><br>Considering any form of user input as untrusted and sanitizing it before operating anywhere.<br><br>Avoid methods such as document.innerHTML and instead use safer functions, for example, document.innerText and document.textContent. If you can, entirely avoid using user input, especially if it affects DOM elements such as the document.url, the document.location, or the document.referrer. |
| **Status** | **OPEN** |

**Supportive evidence (03-Jun-2023):**



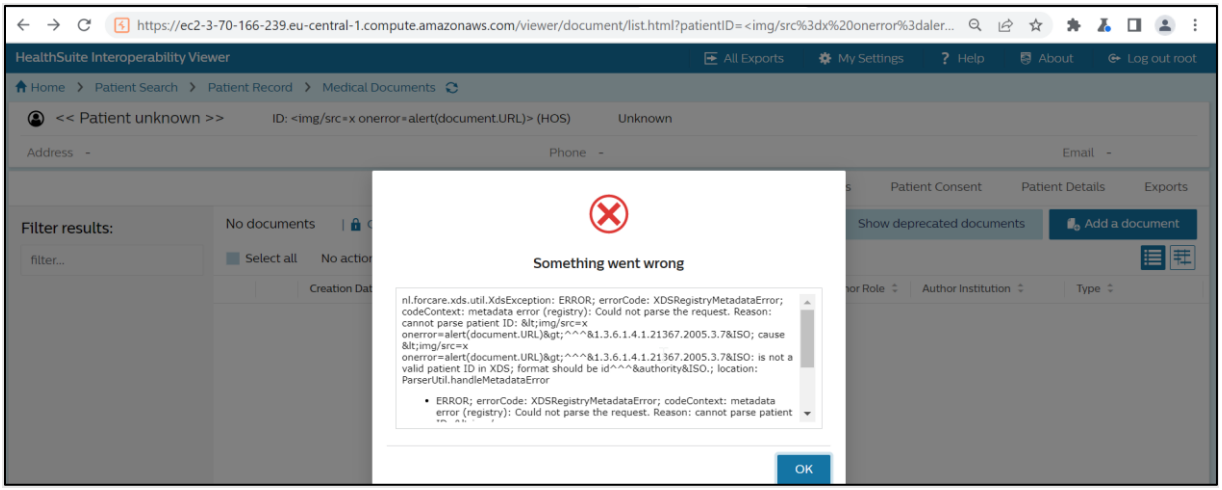

**Supportive evidences:**

**Payload:** <img/src%3dx%20onerror%3dalert(document.URL)>




**<u>Retest (31-Jan-2023):</u>**

Throws uncaught exception error which indicates the fix is not implemented properly.



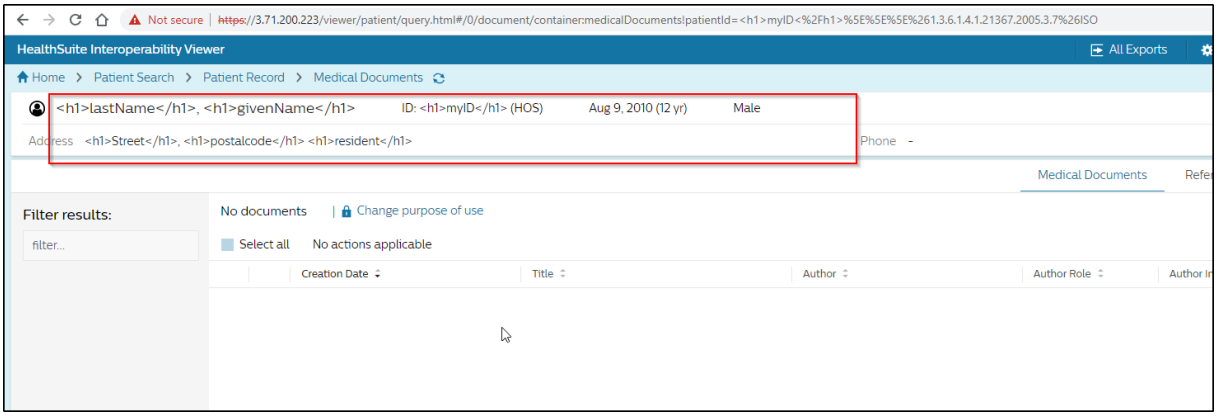
## 8.5 Webapp: Weak Input Validation

| | |
|---|---|
| **Vulnerability Title** | Weak Input Validation |
| **Vulnerability Category** | A3 - Injection |
| **Severity** | **Low** |
| **CVSS V3 Calculation** | CVSS Base Score: 3.1<br>CVSS:3.0/AV:N/AC:H/PR:H/UI:R/S:U/C:N/I:L/A:L |
| **Description** | **Vulnerability Description:** During the assessment it is observed that the application stores or processes untrusted data that is not sufficiently validated. This may be due to a complete lack of validation or validation filters whose implementation does not provide sufficient protection for the given input. An application may obtain data from various external and internal sources including databases, file servers, web services, external client requests, etc. While some of these sources may be considered trustworthy, no assumptions should be made about the validity of data whose source cannot be explicitly verified. This includes not only external data, but also data that was previously stored by the same application and data generated by other entities in the same organization.<br><br>**Retest (06-Jun-2023)**: It is identified that the issue persists.<br><br>**Retest (31-Jan-2023)**: It is identified that the issue is still same like previous time.<br><br>**Exploitability Rational:** Failure to properly validate and handle untrusted input represents the single largest category of software security weaknesses. At a minimum, data that is not validated may impact the application's control flow or data flow, leading to unexpected application states for end users, unintended changes to back-end data, as well as unexpected outcomes from executed application logic.<br><br>An attacker may submit payloads that seek to exploit any number of vulnerabilities that typically result from a lack of input validation. These include (but are not limited to) SQL injection, cross-site scripting, LDAP injection, log injection, and command injection. The consequence of successfully exploiting these vulnerabilities varies, but most provide an attacker with the ability to bypass authentication and/or authorization mechanisms to access, modify or |

| | |
|---|---|
| | delete application and user data, or execute functionality only available to legitimate users.<br><br>**Impact Rational:** An attacker can provide unexpected values and cause a program crash or excessive consumption of resources, such as memory and CPU. An attacker can read confidential data if they cancontrol resource references. An attacker can use malicious input to modify data or possibly alter control flow in unexpected ways, including arbitrary command execution. |
| **Affected Systems/IP Address/URL** | Retest (06-Jun-2023):<br><br>https://server_url/admin/<br><br>https://server_url/viewer/<br><br>Old:<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin/ |
| **Recommendation** | We recommend the following:<br><br>• Data that does not match an expected pattern and data that can potentially be used to execute injection attacks must be discarded or sanitized before use.<br><br>• Perform the validation in such a way that end-users cannot tamper with or bypass the control. Perform the validation on the server-side rather than client-side.<br><br>Whitelist validation should be favored first over other validation techniques since any character or string not explicitly specified as part of the "known-safe" set of characters or values is rejected or removed by default. |
| **Status** | **Open** |

**Supportive evidence**

**Retest (06-Jun-2023):**




**Old POC:**

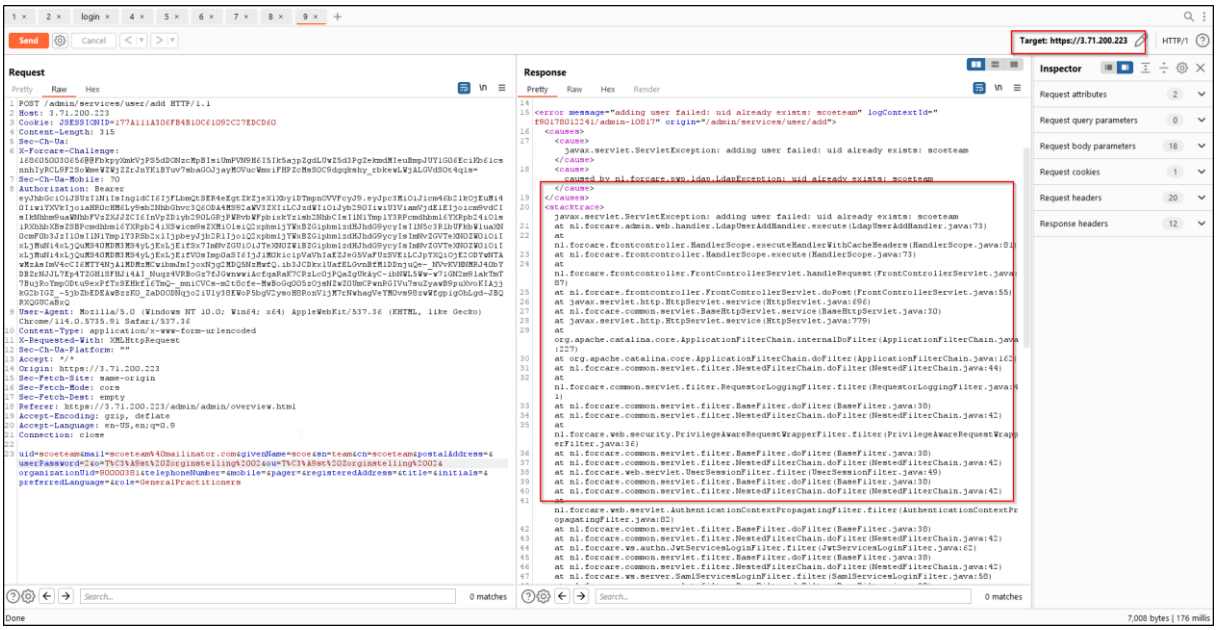
## 8.6 Webapp: Improper Error & Exception Handling

| | |
|---|---|
| **Vulnerability Title** | Improper Error & Exception Handling |
| **Vulnerability Category** | A6 - Security Misconfiguration |
| **Severity** | **Low** |
| **CVSS V3 Calculation** | CVSS Base Score: 3.7<br>CVSS v3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Description** | **Vulnerability Description:** it is observed that the web application reveals sensitive information as part of its error messages such as stack trace, server versions, name of server-side parameter.<br><br>**Retest (06-Jun-2023)**: It is identified that the issue persists.<br><br>**Retest (31-Jan-2023)**: It is identified that same like previous time, errors are displayed to the end user.<br><br>**Exploitability Rational:** An attacker does not require authentication to the platform in order to leverage this vulnerability.<br><br>**Impact Rational:** Attackers may use this vulnerability to gain more information about the system before attempting to attack the web application. |
| **Affected Systems/IP Address/URL** | Retest (06-Jun-2023):<br><br>https://server_url/admin/<br><br>https://server_url/audit/<br><br>https://server_url/viewer/<br><br>Old:<br><br>https:// ec2-3-70-166-239.eu-central-1.compute.amazonaws.com /viewer<br><br>https:// ec2-3-70-166-239.eu-central-1.compute.amazonaws.com /admin/<br><br>https:// ec2-3-70-166-239.eu-central-1.compute.amazonaws.com /audit/ |

| | |
|---|---|
| **Recommendation** | It is recommended to Implement a mechanism to handle and log all errors that pull out the exception stack trace message. It is also recommended to display a generic error message instead of the stack trace. |
| **Status** | **Open** |

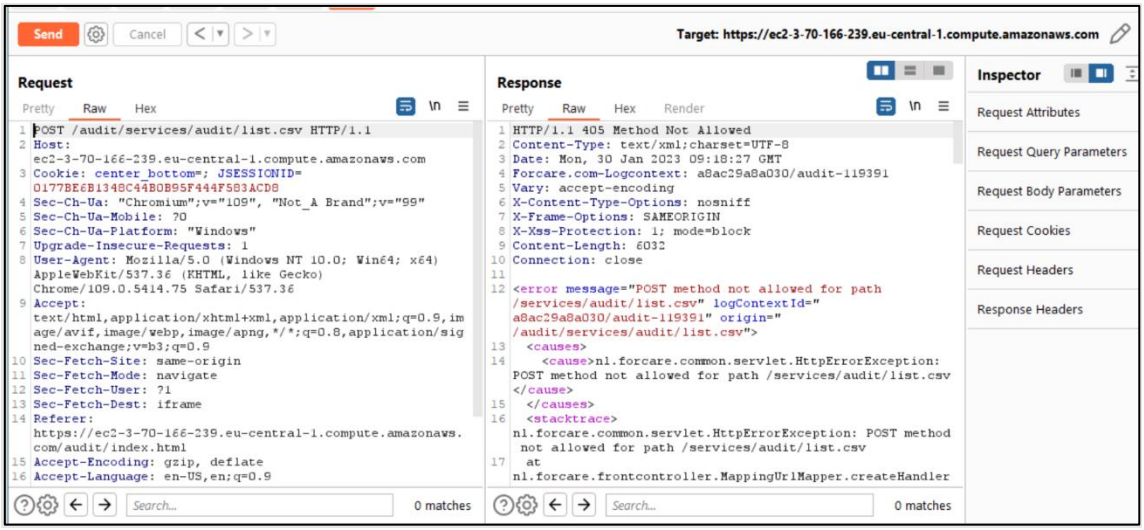
**Supportive evidences**

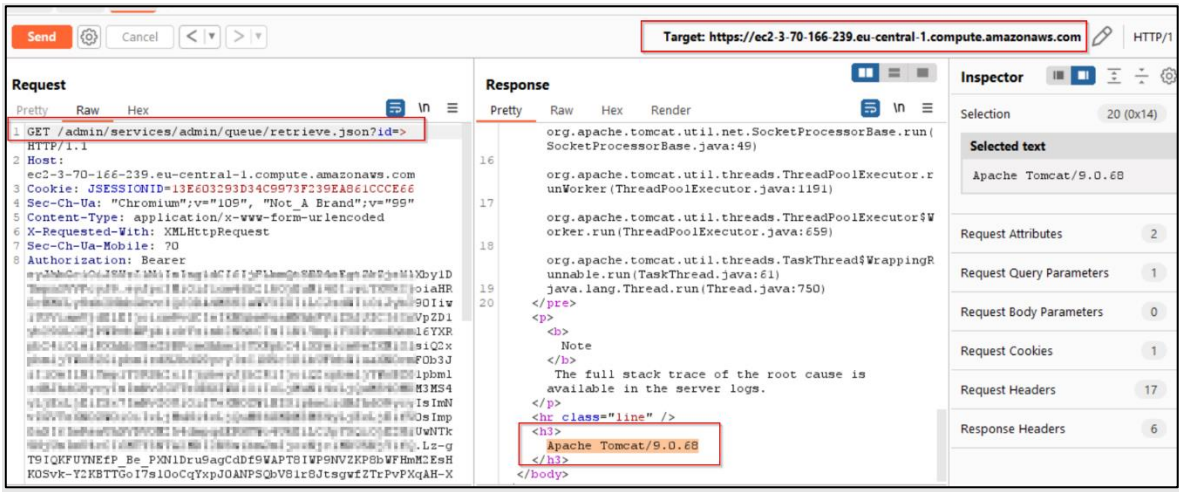
**Retest (06-Jun-2023):**



**Old:**

Improper error and exception as part of server response.



## 8.7 Webapp: Sensitive Information in the URL

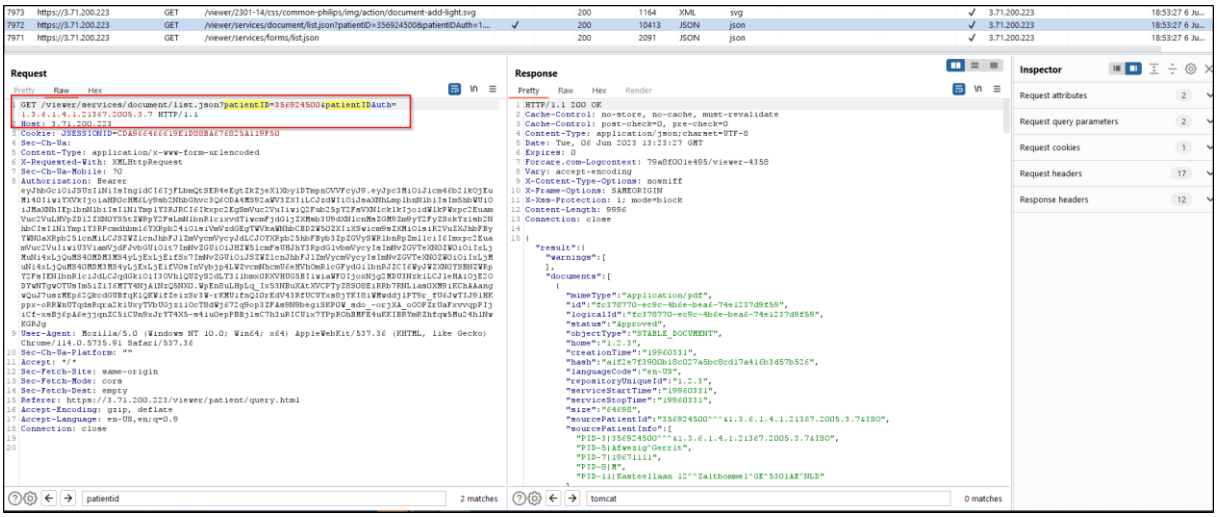
| | |
|---|---|
| **Vulnerability Title** | Sensitive Information in the URL |
| **Vulnerability Category** | A6 - Security Misconfiguration |
| **Severity** | Low |
| **CVSS V3 Calculation** | CVSS Base Score: 3.1<br>CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N |
| **Description** | **Vulnerability Description:** During the security assessment, it is observed that Sensitive information is exposed in via URL query string parameters.Username, PatientID, userID, Code are exposed via URL parameters. URLs may be stored or viewed in multiple places during and after a request is made to the server:<br>• URLs are often logged in multiple places including the browser history, proxy logs, and web server logs.<br>• The query string will be sent as part of the URL if the URL is passed to another site via the Referrer header.<br>• URLs sent to the user as part of an HTML page may be cached on disk.<br><br>**Retest (06-Jun-2022)**: During the retest, it is identified that the URL still contains sensitive information.<br><br>**Retest (31-Jan-2022)**: It is identified that the URL still contains sensitive information.<br><br>**Exploitability Rational:**<br>Potential access vectors may include, but are not limited to:<br>• Browser history, proxy logs, web server logs, etc.<br>• Utilizing other attacks (such as cross-site scripting) to extract sensitive information from the source of a page containing links to URLs with sensitive information in the query string<br>• Shoulder-surfing the URL in a user's browser address bar. |

| | |
|---|---|
| | **Impact Rational: A**ttacker who gains access to any location where URLs are stored can view sensitive information passed via the query string. Depending on the nature of the information, a malicious user may obtain personally identifiable information (PII), private user data or information which would allow user impersonation (in the event of credential or session identifier. |
| **Affected Systems/IP Address/URL** | Retest (06-Jun-2023):<br><br>https://server_url/viewer/services/document/list.json?patientIDAuth=<Auth_ID>&patientID=<PatientID><br><br>Old:<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/services/document/list.json?patientIDAuth=1.3.6.1.4.1.21367.2005.3.7&patientID=134442134 |
| **Recommendation** | Do not pass the sensitive data like credentials, UserID, codes or sessionIDs between the client and server via URL query string parameters. |
| **Status** | **Open** |

**Steps to Reproduce**

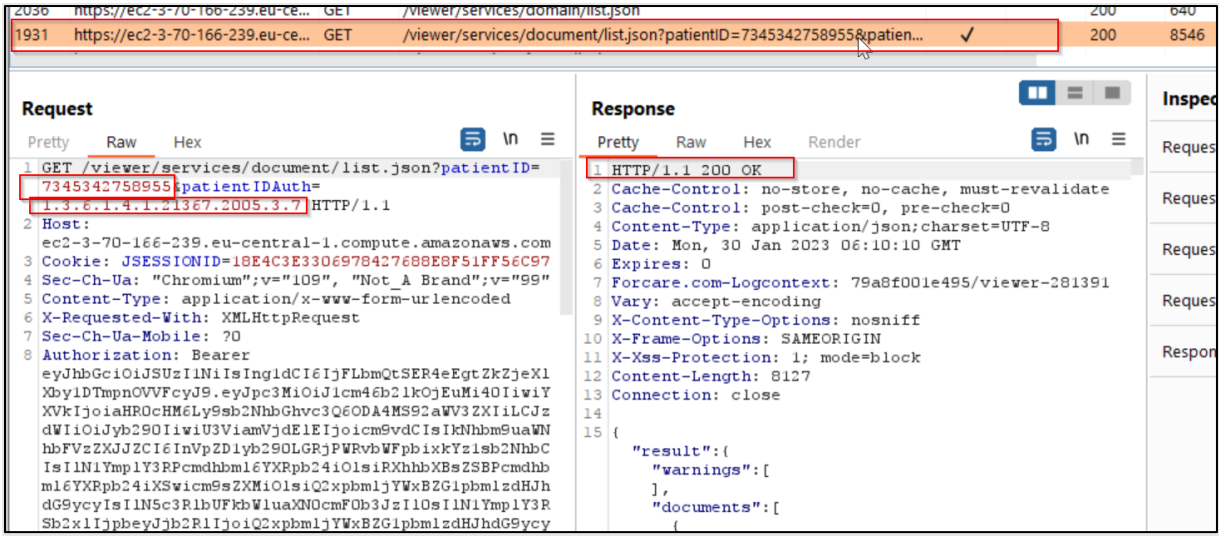
**Retest (06-Jun-2023):**

Steps are same as before.



**Old POC:**

- Launch ForView Applications.
- Intercept the requests using Burp proxy tool.
- As observed in below request PatientID and PatientIDAuth value gets sent for GET requests, which can get logged.



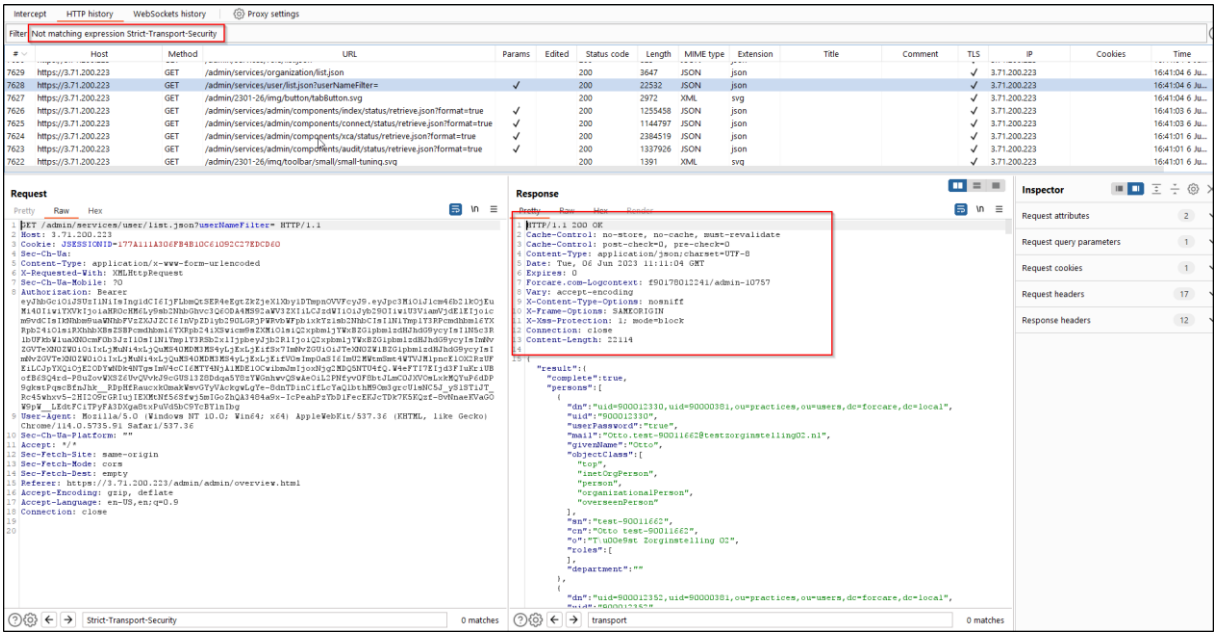
## 8.8 Webapp: HTTP Strict Transport Security (HSTS) Not Implemented

| | |
|---|---|
| **Vulnerability Title** | HTTP Strict Transport Security (HSTS) Not Implemented |
| **Vulnerability Category** | A6 - Security Misconfiguration |
| **Severity** | **Low** |
| **CVSS V3 Calculation** | CVSS Base Score: 3.8<br>CVSS:3.0/AV:L/AC:H/PR:H/UI:R/S:U/C:L/I:L/A:L |
| **Description** | **Vulnerability Description:**<br><br>The server does not implement the "HTTP Strict-Transport Security" (HSTS) web security policy mechanism. When HSTS is enabled, the web application sends a special response header, "Strict-Transport-Security" to the client with a duration of time specified. Once a supported browser receives this header, that browser can only make requests to the application over HTTPS for the duration of time specified in the header. Any links to resources over HTTP will be rewritten to HTTPS before the request is made.<br><br>**Retest (06-Jun-2023)**: It is identified that the issue persists.<br><br>**Retest (31-Jan-2022)**: It is identified that just like previous time issues is same.<br><br>**Exploitability rational**<br><br>Applications that do not utilize the "HTTP Strict-Transport Security" policy are more susceptible to man-in-the-middle attacks via SSL stripping, which occurs when an attacker transparently downgrades a victim's communication with the server from HTTPS to HTTP. Once this is accomplished, the attacker will gain the ability to view and potentially modify the victim's traffic, exposing sensitive information and gaining access to unauthorized functionality.<br><br>Attacker can gain sensitive information and access for the unauthorized functionality. |

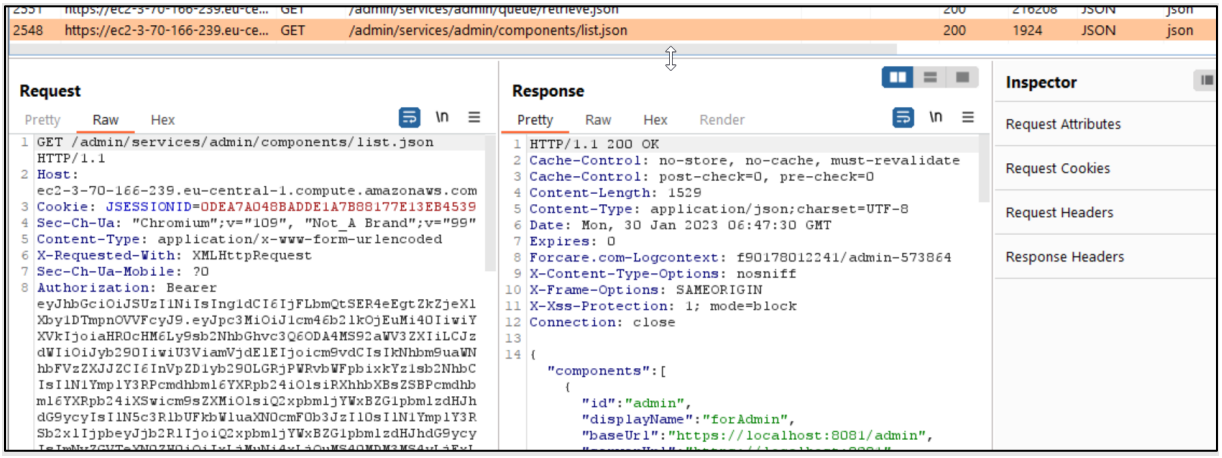
| | |
|---|---|
| | **Impact rational:** Attacker can gain sensitive information and access for the unauthorized functionality. |
| **Affected Systems/IP Address/URL** | Retest (06-Jun-2023):<br>https://server_url/admin/<br>https://server_url/audit/<br>https://server_url/viewer/<br>https://server_url/imageupload/<br>Old:<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/audit<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/imageupload?assigningAuthority=1.3.6.1.4.1.21367.2005.3.7#/start<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer |
| **Recommendation** | The application server should send the "Strict-Transport-Security" HTTP header in each response indicating that future requests to the domain use only HTTPS. The following is a basic example of the HSTS HTTP header, setting a max-age of one year:<br>Strict-Transport-Security: max-age=31536000<br>Subdomains should also be configured in this manner, by including the "includeSubDomains" flag:<br>Strict-Transport-Security: max-age=31536000; includeSubDomains; |
| **Status** | **Open** |

**Steps to Reproduce**

**Retest (06-Jun-2023):**




**Old:**



## 8.9 Webapp: No Account Lockout Policy

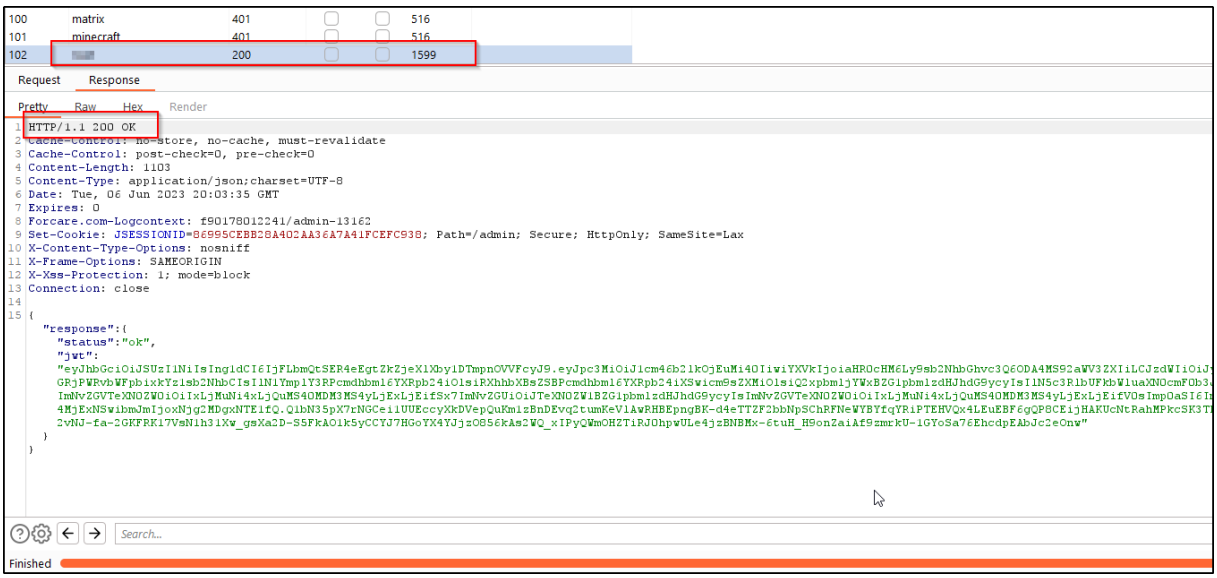
| | |
|---|---|
| **Vulnerability Title** | No Account Lockout Policy |
| **Vulnerability Category** | A2 - Broken Authentication |
| **Severity** | **Low** |
| **CVSS V3 Calculation** | CVSS Base Score: 3.7<br>CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L |
| **Description** | **Vulnerability Description:**<br><br>The application does not maintain or enforce an account lockout policy. A lockout policy refers to the mechanism that temporarily suspends a user's account after a certain number of unsuccessful authentication attempts have been made.<br><br>**Retest (06-Jun-2023)**: It is identified that the issue persists.<br><br>**Retest (31-Jan-2022)**: It is identified that just like previous time issues is same.<br><br>**Exploitability rational:** Applications with no lockout policy are vulnerable to brute force password guessing attacks, in which the attacker performs login attempts using a known username and a list of potential passwords until a successful combination is found. Once a successful combination is discovered, the attacker is granted full access to the compromised account, and can impersonate the victim without detection.<br><br>**Impact rational:** Without a strong lockout mechanism, the application can be susceptible to brute force attacks. |
| **Affected Systems/IP Address/URL** | Retest (06-Jun-2023):<br><br>https://server_url/admin/<br><br>https://server_url/audit/ |

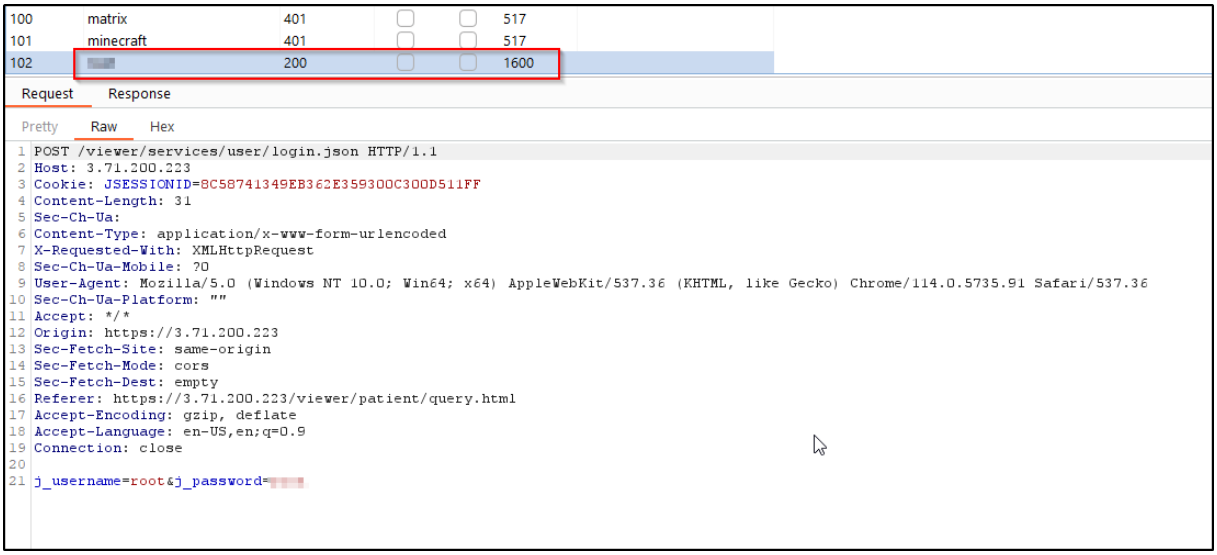
| | |
|---|---|
| | https://server_url/viewer/<br>Old:<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/admin/<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/audit/ |
| **Recommendation** | It's recommended to implement some type of account lockout after a defined number of incorrect password attempts. |
| **Status** | **Open** |

**Steps to Reproduce**

**Retest (06-Jun-2023):**

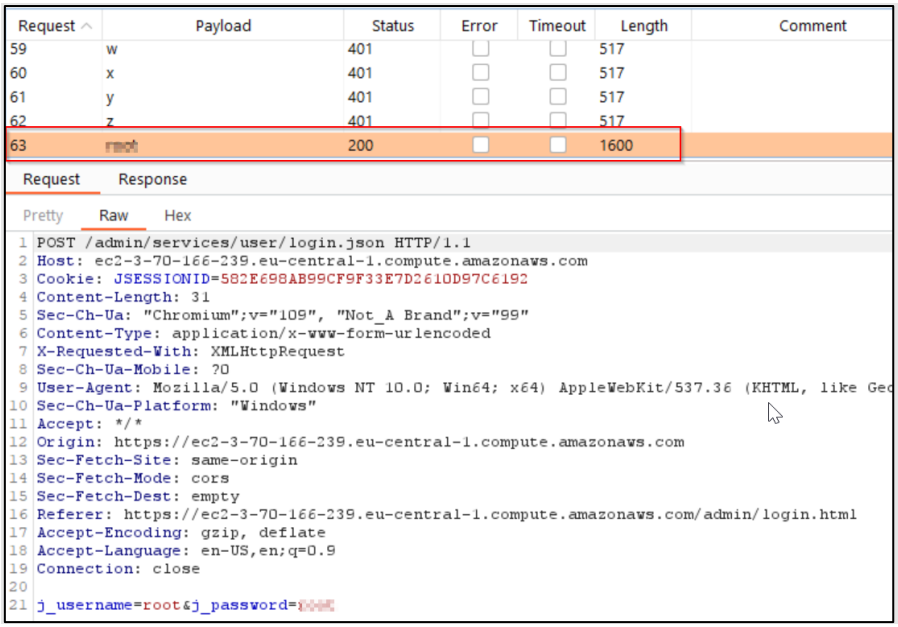
Steps are same as before.






**Old POC:**

- Login with username and password multiple times then you will be able to see that the account does not get locked out.



## 8.10 Webapp: Reflected Cross-Site Scripting (XSS)

| | |
|---|---|
| **Vulnerability Category** | Reflected Cross-Site Scripting (XSS) |
| **Vulnerability Category** | A3- Injection |
| **Severity** | **Low** |
| **CVSS V3 Calculation** | CVSS Base Score: 3.5<br>CVSS:3.0/AV:N/AC:L/PR:H/UI:R/S:U/C:L/I:L/A:N |
| **Description** | **Vulnerability Description:** A Reflected Cross-Site Scripting (XSS) vulnerability occurs when a web application sends strings that are provided by an attacker to a victim's browser in such a way that the browser executes part of the string as code. The string contains malicious data and is passed to the application through a parameter that an attacker can control (For example, a URL parameter or an HTML form field). The application immediately inserts it into its response. This results in the victim's browser executing the attacker's code within a legitimate user's session. Attackers typically exploit reflected XSS vulnerabilities by sending users malicious links containing JavaScript code (For example, via e-mail) or by posting malicious code to other sites that the vulnerable application's users may visit.<br><br>**Retest (06-Jun-2023)**: It is identified that the issue persists.<br><br>**Retest (31-Jan-2023)**: It is identified that self-XSS is still possible & exploitable in multiple places.<br><br>**Exploitability rational:** Reflected Cross-Site Scripting vulnerabilities give the attacker control of the user's browser. The attack can alter page content with malicious HTML or JavaScript code. The attacker can arbitrarily alter page content displayed to the victim and can execute application functions using the victim's application identity if the victim is authenticated to the application. An often cited example use of a Reflected Cross-Site is where the attacker send themselves to the victim's session identifier. With this session identifier, the attacker can then perform application functions using that user's identity for the duration of that session. |

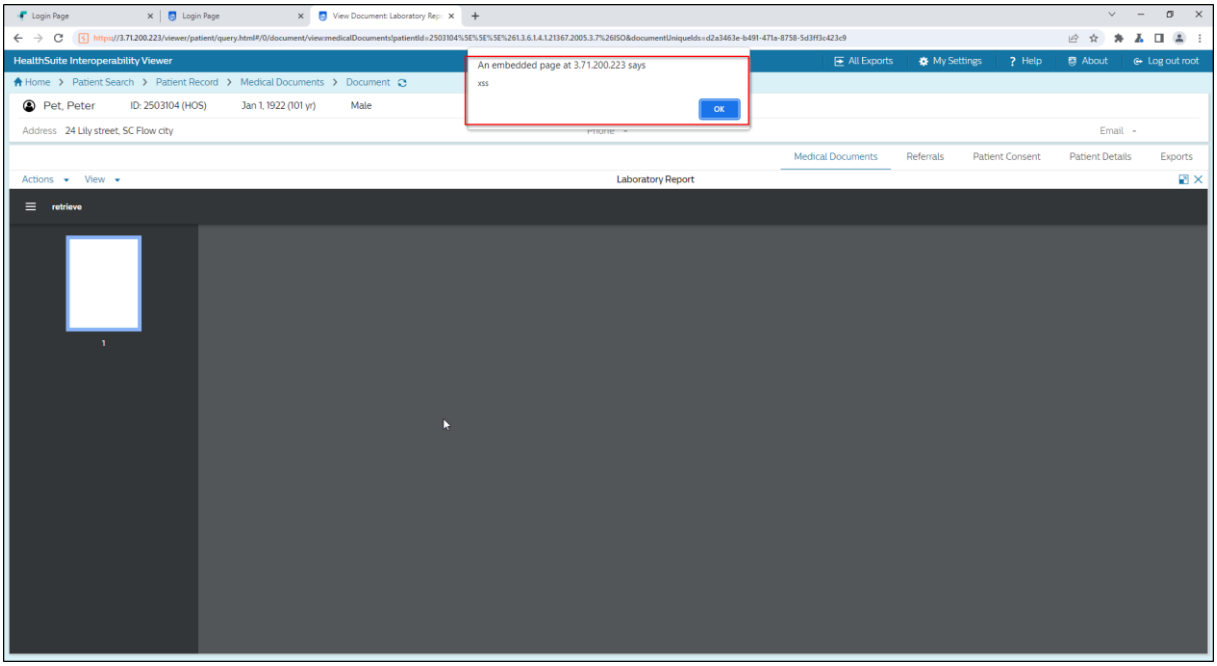
| | |
|---|---|
| | **Impact rational:** Attackers can steal cookies and change the temp look and feel using reflected cross site scripting. |
| **Affected Systems/IP Address/URL** | Retest (06-Jun-2023) :<br><br>https://server_url/viewer/patient/query.html#/0/document/view:medicalDocuments!patentId=<patientID><PatentAuthID>ISO&documentUniqueIds=<documentID><br><br>Old:<br><br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com/viewer/services/connect/proxy/flow/sendOruWithPdf |
| **Recommendation** | Reflected Cross-Site Scripting (XSS) is prevented by encoding data before inserting it into the generated web page. Each character of the data is encoded and the result string is then inserted onto the generated web page. This technique of encoding values before inserting them on the web page is called "Output Encoding". Output encoding libraries exist for most popular programming languages and frameworks.<br><br>A web page has seven different output contexts and each output context requires a different encoding scheme. Encode the data using the proper scheme. The seven different encoding schemes are:<br><br>• HTML Text Element<br>• HTML Attribute<br>• URL Parameter<br>• JavaScript Literal<br>• HTML Comment<br>• HTTP Header<br>• CSS Property<br><br>For example, the characters: <, >, ", ' are encoded as <, >, ", ' for when those characters are inserted into an HTML Text Element. When those characters are inserted as a URL Parameter, the same characters are encoded as %3C, %3E, %22, %27. |

| | |
|---|---|
| | Libraries for implementing the encoding schemes exist for most popular programming languages.<br>• OWASP Java Encoder: Java only<br>• Microsoft Web Protection Library: .NET languages<br>• Ruby – escapeHTML() - only supports HTML Text Encoding<br>• Jgencoder in JQuery: for preventing DOM-based XSS<br>Green field projects can consider the use of other technologies:<br>• Google Capabilities based JavaScript CAJA<br>• OWASP JXT– automatically encodes string data with the proper encoding.<br><br>Input validation is often recommended as a way to mitigate reflected cross-site scripting. It is insufficient, however, because input validation isused to prevent cross-site scripting only when the data has a strict syntactic format, such as numeric values and dates. Any application inputs which must accept arbitrary data would remain vulnerable. |
| **Status** | **OPEN** |

**Steps to Reproduce:**

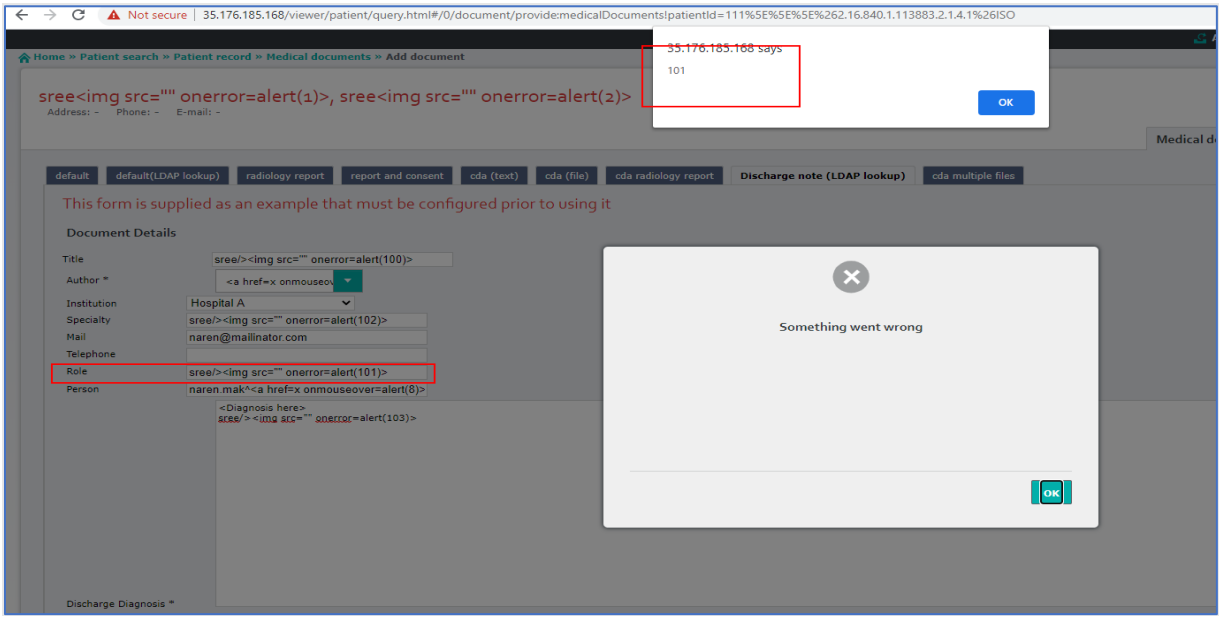
**Retest (06-Jun-2023):**

Steps are same as mentioned in 'Retest Status( as on 30 Jan 2023)'.



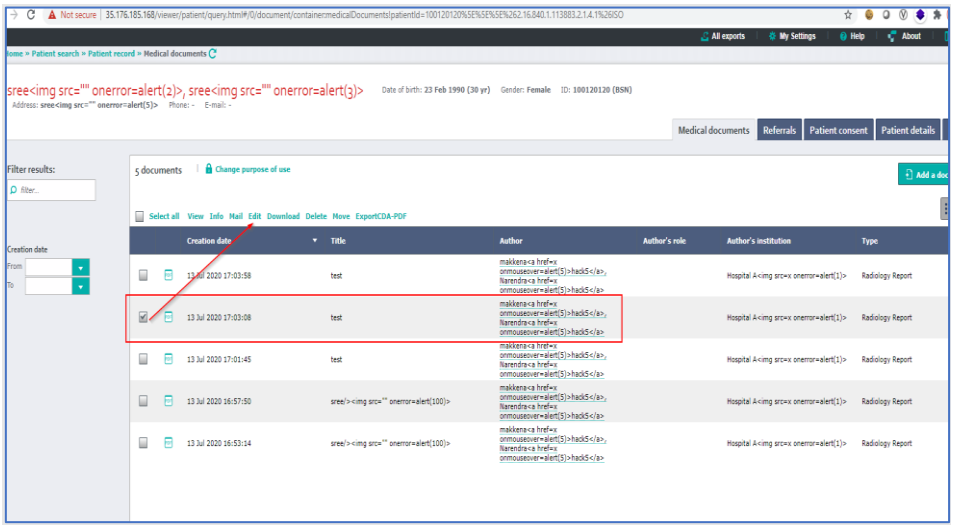

**Old POC:**

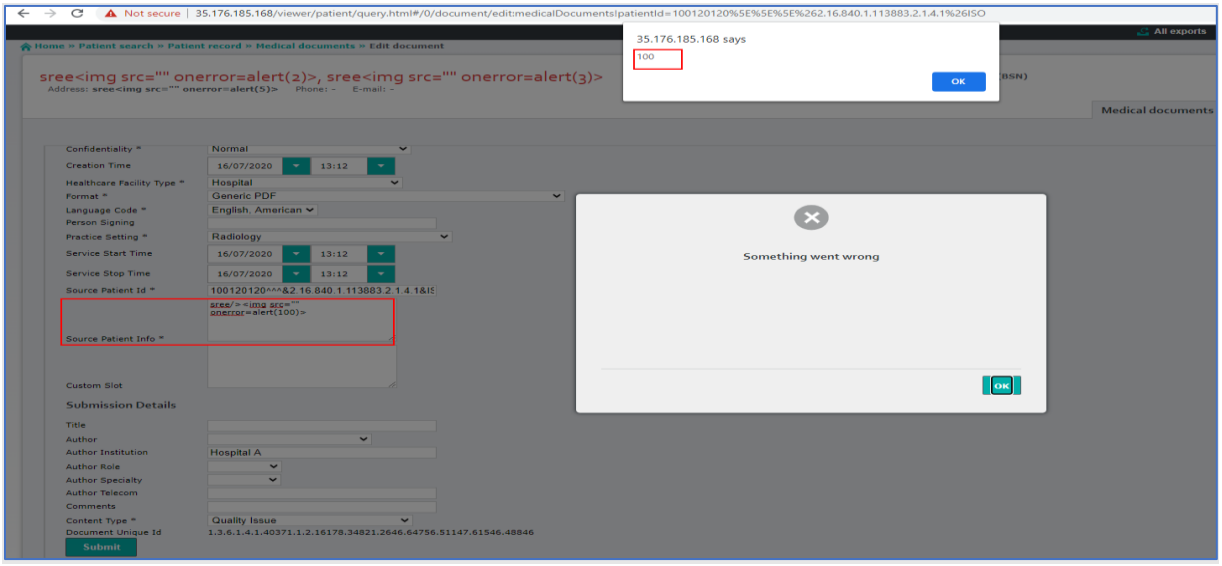
Login to Viewer > Patient search >> select any patient >>> medical documents >>>> Add document >>>>> Discharge Note(Ldap lookup) >>>>>> fill the form as shown in the below snapshot and then submit, the application server responds with same malicious script without validation or encoding and execute at browser.



Login to Viewer > Patient search >> select any patient >>> medical documents >>>> select any xml document >>>>> actions >>>>>> edit

Parameters: Source Patient ID & Source patient Info

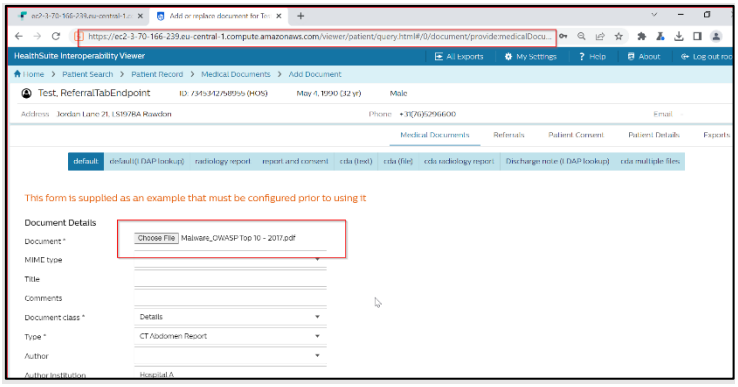


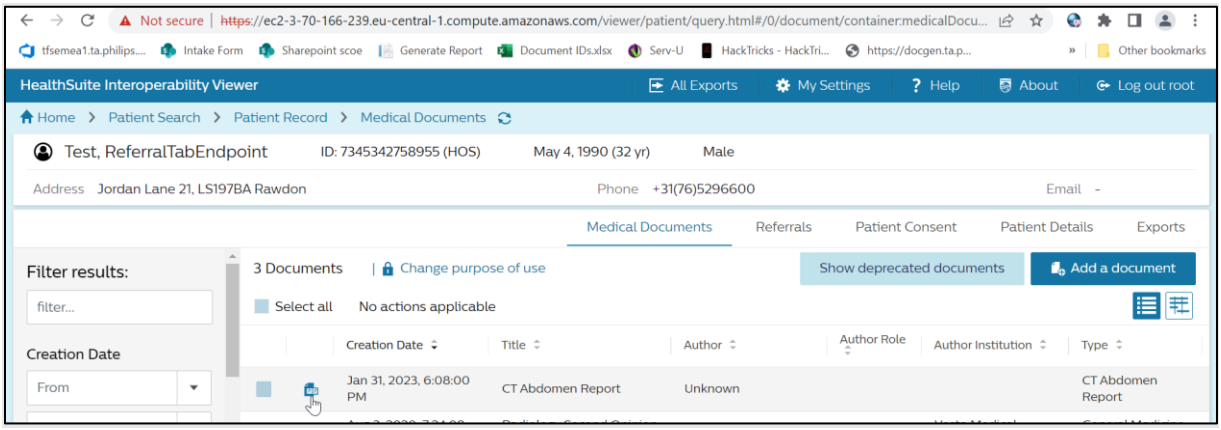

**Retest Status( as on 30 Jan 2023)**

Issue still persists.

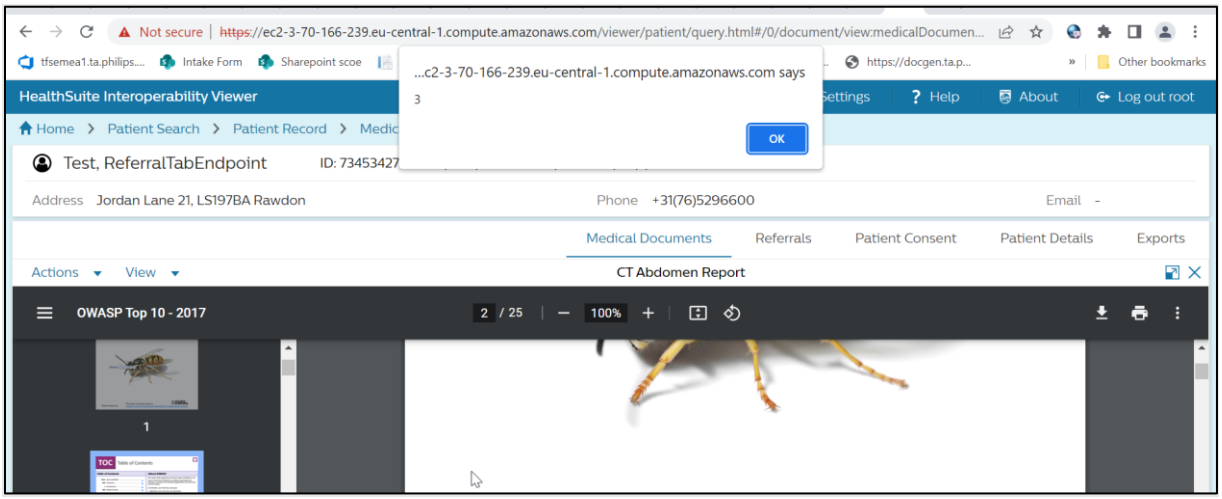
Uploading a pdf which has malicious script.




- Upload the file and navigate to Medical Documents section.
- Click the pdf icon for the patient where the document is upload.




- Scroll the launched pdf and you can see the javascript getting executed.



## 8.11 Webapp: Weak SSL/TLS Configuration

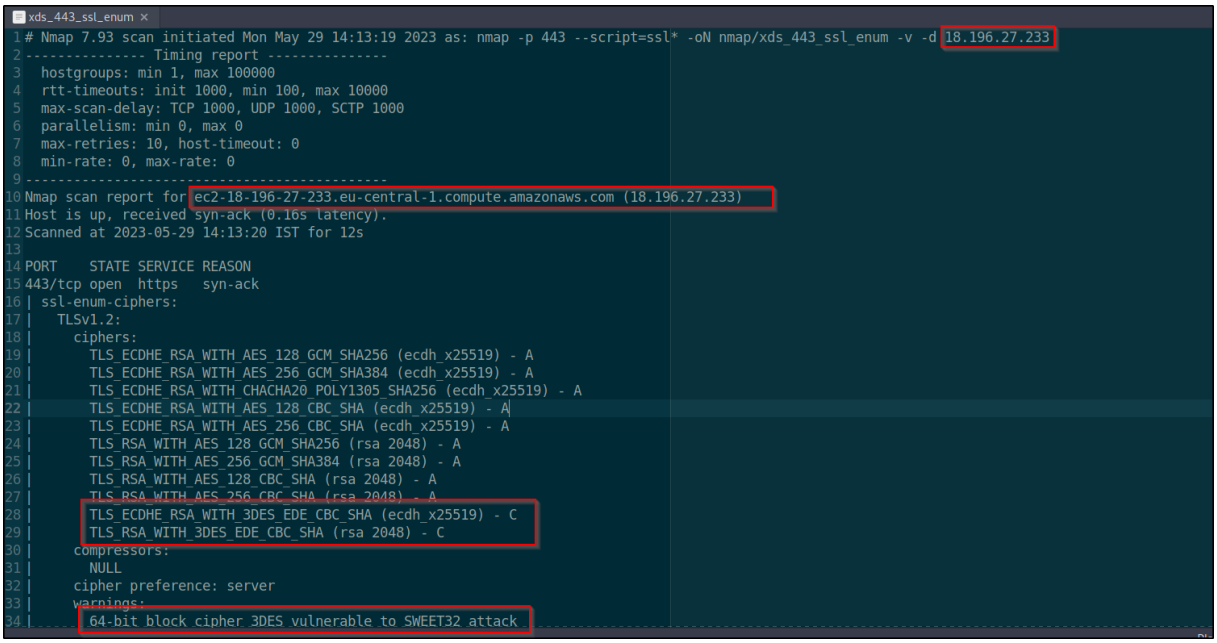
| | |
|---|---|
| **Vulnerability Title** | Weak SSL/TLS Configuration |
| **Vulnerability Category** | A2 Cryptographic Failures |
| **Severity** | **Low** |
| **CVSS V3 Calculation** | CVSS Base Score: 3.1<br>CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N |
| **Description** | **Vulnerability Description:** It is observed that the server-side SSL/TLS endpoint is configured to allow weak SSL/TLS cipher suites in the provided server.<br>TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048)<br>TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048)<br>**Retest (06-Jun-2023)**: It is identified that the issue persists.<br>**Exploitability Rational:** Some misconfiguration in the server can be used to force the use of a weak cipher - or at worst no encryption - permitting to an attacker to gain access to the supposed secure communication channel. Other misconfiguration can be used for a Denial of Service attack.<br>**Impact Rational:** A server-side SSL/TLS endpoint that supports weak ciphers can allow an attacker to read or modify traffic sent in SSL/TLS connections with that endpoint. |
| **Affected Systems/IP Address/URL** | Retest (09/06/2023):<br>https://server_url/<br>Old:<br>https://ec2-3-70-166-239.eu-central-1.compute.amazonaws.com |
| **Recommendation** | Update the server-side TLS endpoint's configuration to allow only TLSv1.2 or TLSv1.3 connections with cipher suites that use the following: |

| | |
|---|---|
| | • Ephemeral Diffie-Hellman for key exchange (Optionally, allow RSA for key exchange if necessary for supporting some clients).<br>• Block ciphers in GCM mode. Note: If CBC mode must be allowed for supporting some clients, use only CBC mode cipher suites that use the SHA2 family of hash functions (SHA256, SHA384, SHA512). |
| **Status** | **Open** |

**Steps to Reproduce**

**Retest (06-Jun-2023):**

Steps are same as before.




**Old POC:**

Use nmap to enumerate the ciphers used by the application end-points.

```
Starting Nmap 7.92 ( https://nmap.org ) at 2023-01-23 18:30 India Standard Time
Nmap scan report for ec2-3-70-166-239.eu-central-1.compute.amazonaws.com (3.70.166.239)
Host is up (0.14s latency).

PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (ecdh_x25519) - C
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.3:
|     ciphers:
|       TLS_AKE_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_AKE_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_AKE_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|     cipher preference: server
|_  least strength: C

Nmap done: 1 IP address (1 host up) scanned in 12.22 seconds
```

## 9. Tools Used

| Scope | Tools Used |
|---|---|
| Application Security | Burpsuite pro, nmap |

## 10. Automated Tool Report

NA

# 11. Manual Test Reports and Test Case Execution

Testsuite_WebApp.xlsx