OpenMRS

MEDICAL RECORD SYSTEM

OpenMRS

The Squadratics

Patrick Lindsay, Mittranj Pansuriya, Travis Ingram, Jeson Rijal

OUTLINE

- Project Description
- Reflections on Assurance Claims
- Gaps in Security Requirements and Design
- Findings from Manual Code Review
- Findings from Automated Code Scanning
- Contribution to the original project
- Summary / Conclusions
- Sources and Links

OpenMRS

- OpenMRS is a software platform and a reference application which enables design of a customized medical records system
- Patient-based medical record system focusing on giving providers a free customizable electronic medical record system (EMR)
- It is based on the principle that information should be stored in a way which makes it easy to summarize and analyze
- Modular architecture
  - Allows developers to extend the OpenMRS core functionality by creating modules that can easily be added or removed to meet the needs of a specific implementation.

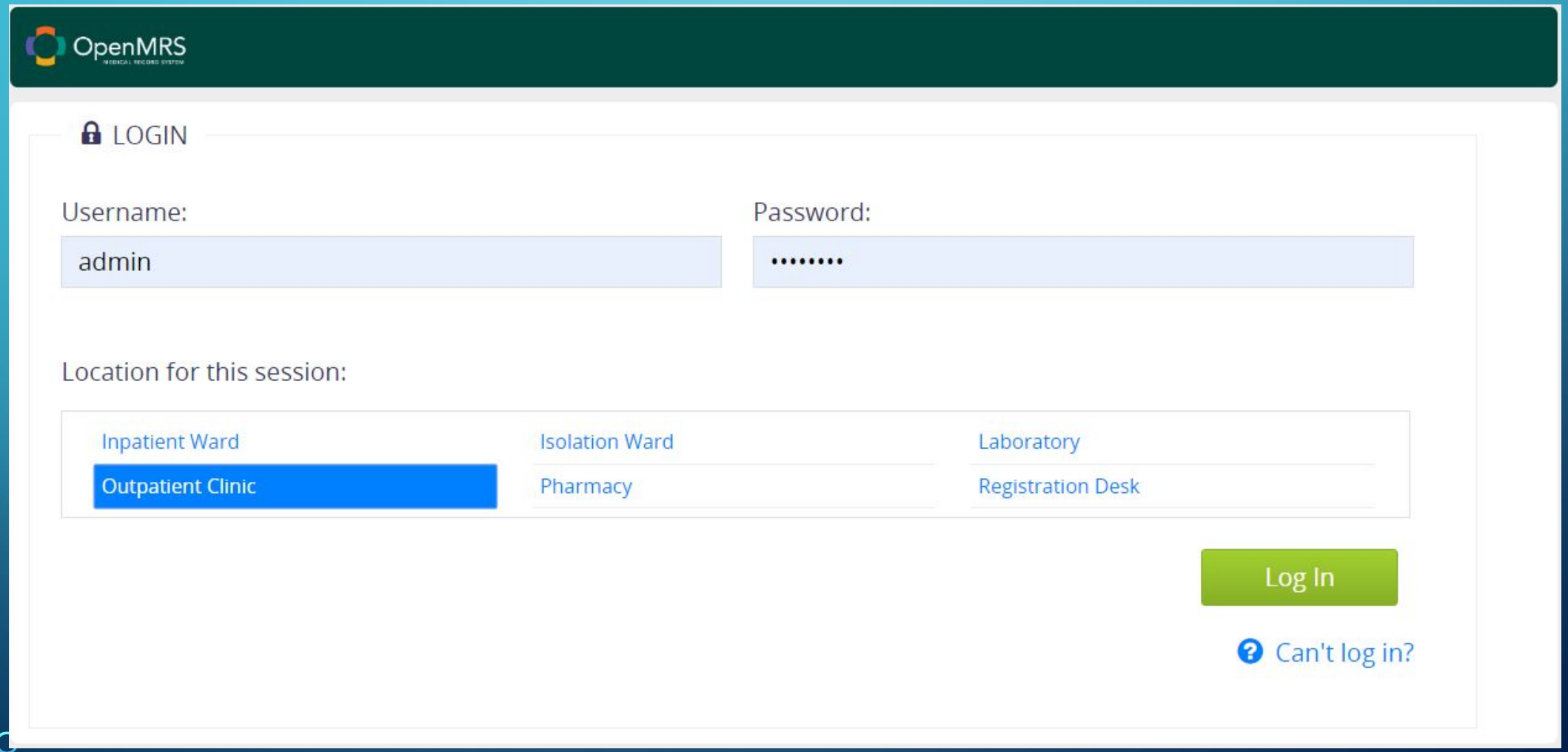
POPULARITY AND CODEBASE

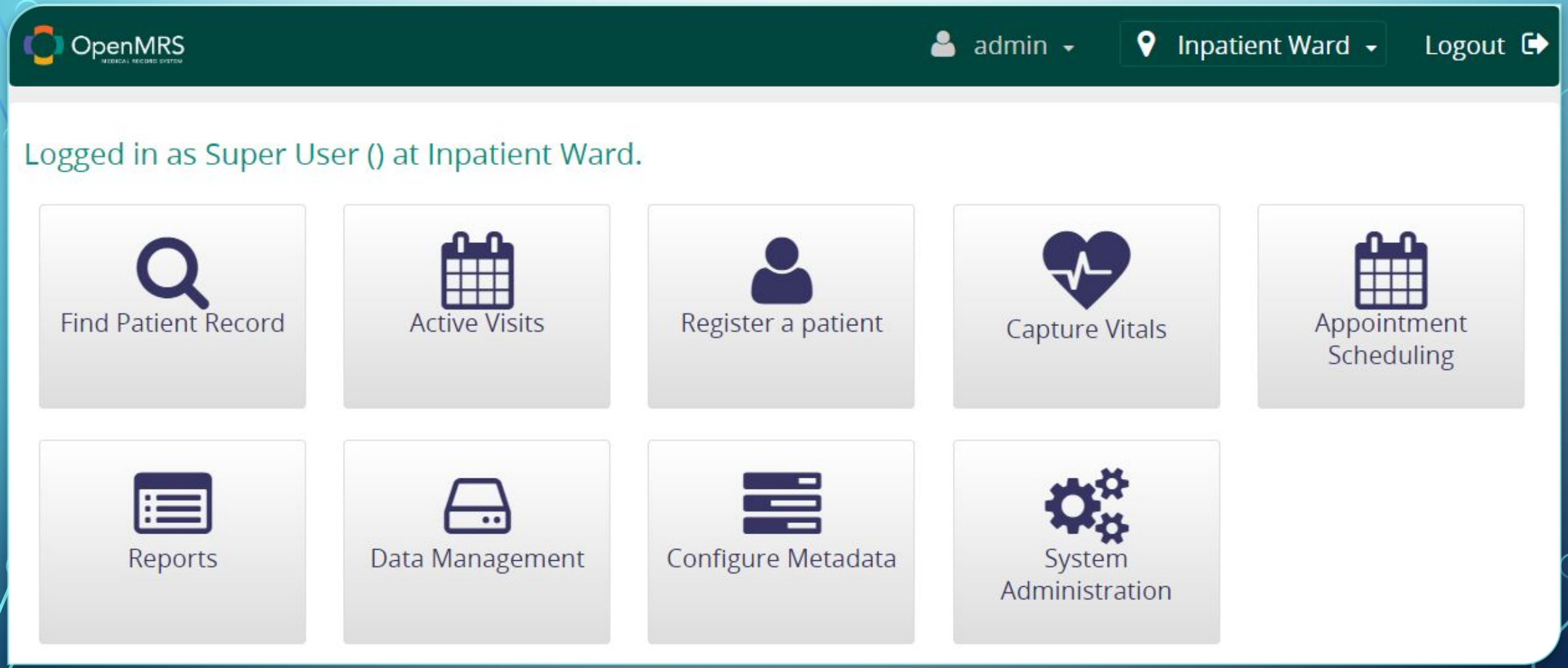
- Contributors: 324 direct GitHub contributors
- Activity:
  - 140 releases
  - 10,803 commits (with the most recent commit coming less than 3 hours ago)
  - 72 pending pull requests
  - 43 branches
  - 2,413 forks
- Use and Popularity: OpenMRS is used in almost 50 deployments across 6 continents with uses spanning from research and development to evaluation and clinical
- Languages Used:
  - Java – 96.2%
  - SQLPL – 2.9%
  - other languages – <1%

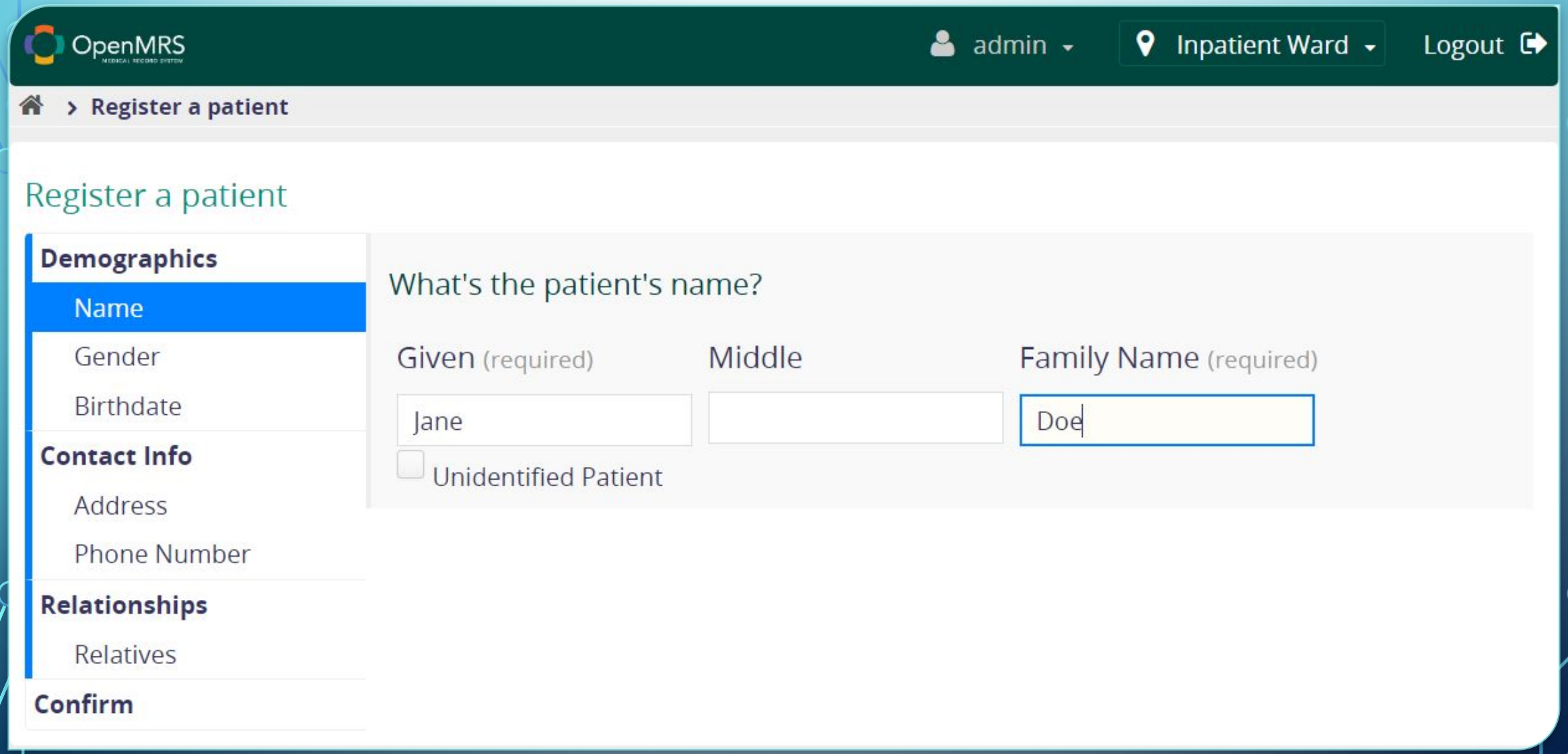
FEATURES

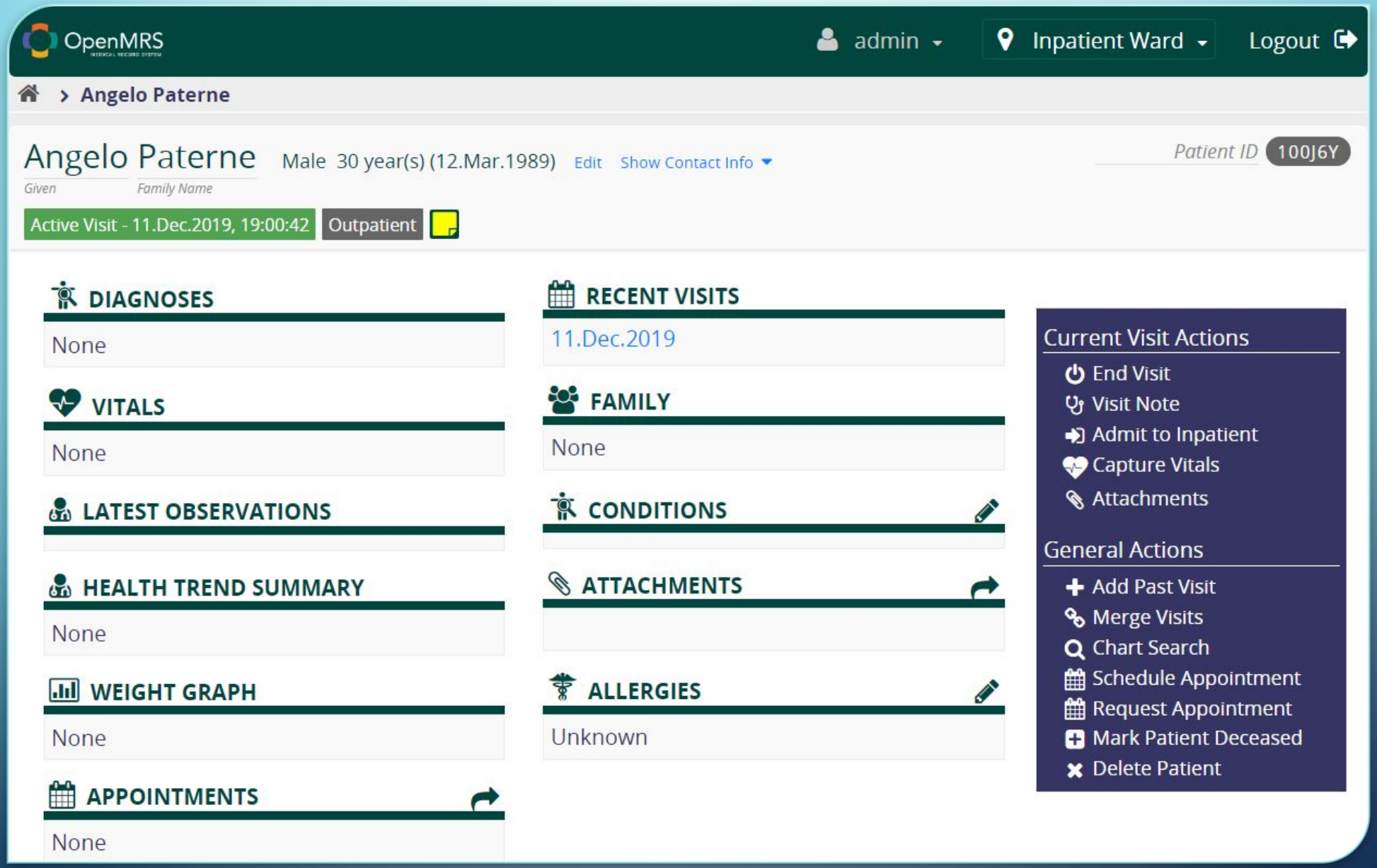
- Privilege-based access
  - User roles and permission system
- Central concept dictionary
  - Definitions of all data (both questions and answers) are defined in a centralized dictionary, allowing for robust, coded data
- Patient repository
  - Creation and maintenance of patient data, including demographics, clinical observations, encounter data, orders, etc.
- Localization / internationalization
  - Multiple language support and the possibility to extend to other languages with full UTF-8 support
- Support for complex data
  - Radiology images, sound files, etc. can be stored as “complex” observations

DEMO
(OPERATIONAL ENVIRONMENT)




OpenMRS
admin
Inpatient Ward
Logout
Logged in as Super User () at Inpatient Ward.
Find Patient Record
Active Visits
Register a patient
Capture Vitals
Appointment Scheduling
Reports
Data Management
Configure Metadata
System Administration


OpenMRS
admin
Inpatient Ward
Logout
Register a patient
Register a patient
Demographics
Name
Gender
Birthdate
Contact Info
Address
Phone Number
Relationships
Relatives
Confirm
What's the patient's name?
Given (required)
Middle
Family Name (required)
Jane
Doe
Unidentified Patient


OpenMRS
admin
Inpatient Ward
Logout
Angelo Paterne
Angelo Paterne
Male 30 year(s) (12.Mar.1989)
Edit
Show Contact Info
Given
Family Name
Patient ID
100J6Y
Active Visit - 11.Dec.2019, 19:00:42
Outpatient
DIAGNOSES
None
VITALS
None
LATEST OBSERVATIONS
HEALTH TREND SUMMARY
None
WEIGHT GRAPH
None
APPOINTMENTS
None
RECENT VISITS
11.Dec.2019
FAMILY
None
CONDITIONS
ATTACHMENTS
ALLERGIES
Unknown
Current Visit Actions
End Visit
Visit Note
Admit to Inpatient
Capture Vitals
Attachments
General Actions
Add Past Visit
Merge Visits
Chart Search
Schedule Appointment
Request Appointment
Mark Patient Deceased
Delete Patient

REFLECTION ON ASSURANCE CLAIM

- Assurance Claim 1: OpenMRS is resilient to XSS attacks
- Assurance Claim 2: OpenMRS is adequately secure against network eavesdropping
- Assurance Claim 3: The file upload functionality is resilient to arbitrary code execution (further discussion below)
- Assurance Claim 4: OpenMRS is acceptably secure against Authentication Abuse (further discussion below)
- Assurance Claim 5: OpenMRS is acceptably secure against unauthorized access/login to the software (further discussion below)

REFLECTION ON ASSURANCE CLAIM

- Assurance Claim 1: Cross-site Scripting
  - Misuse - Compromise through malicious file injection
  - Prevention
    - No client side scripting
    - Strong session implementation
    - Encode HTML contents and attributes
  - We found this acceptable

REFLECTION ON ASSURANCE CLAIM

- Assurance Claim 2: Network eavesdropping
  - Misuse - Network compromise via unrestricted access
  - Prevention - Access control lists for individual users
  - We found this acceptable

REFLECTION ON ASSURANCE CLAIM

- Assurance Claim 3: File Upload
  - Misuse - Crafting a malicious file to upload reverse shell, duplicate file extension (file.txt.jpg)
  - Prevention – Whitelisting files based on file extension, limiting the types of file able to be upload to the system.
  - Caveat - Requires proper configuration / administration

REFLECTION ON ASSURANCE CLAIM

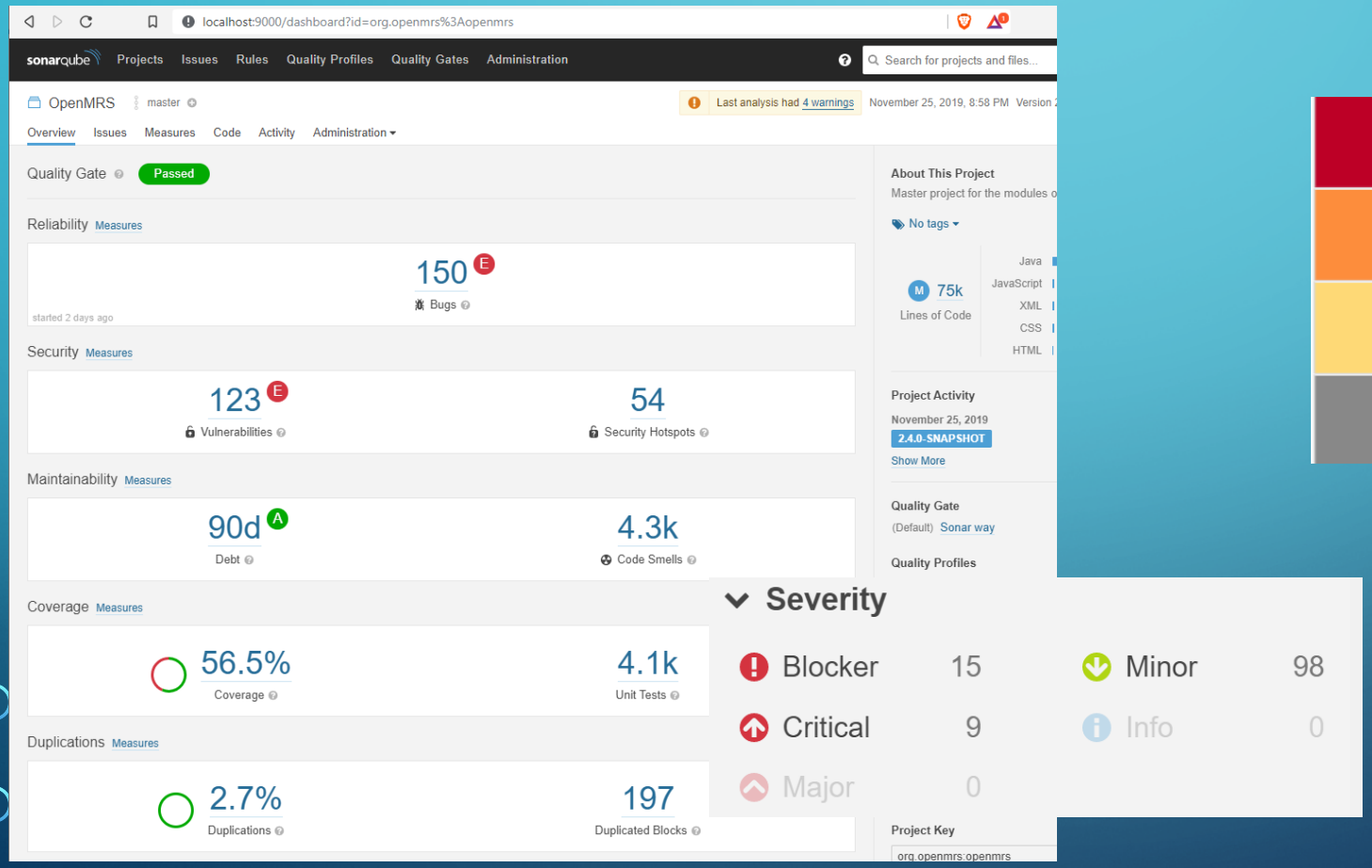
- Assurance Claim 4/5: Authentication
  - Misuse – Make multiple login attempts
  - Prevention – Brute-force attack can be prevented by using strategy like password complexity, password length, two factor authentication, limiting login attempts.
  - Additional features – Two-way encryption, hash matching which is mostly helpful with password validation and checks against both SHA 1 and SHA-512 + 128-character set algorithms.
  - Caveat - Cryptographic algorithm could be improved with GCM

GAPS IN SECURITY REQUIREMENTS AND DESIGN

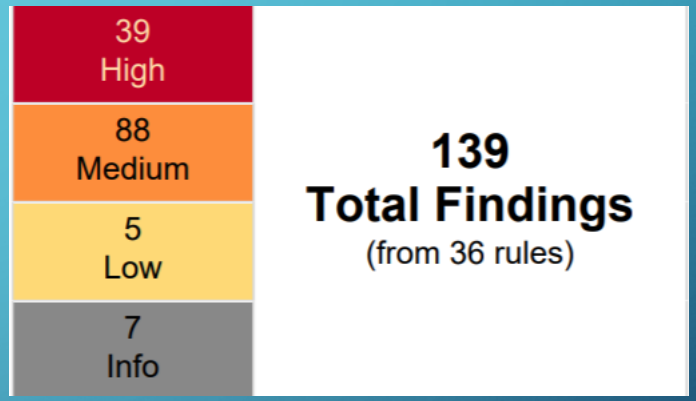
- Gaps Summary
  - Doesn’t have feature of limiting login attempts
  - Logged in user's connection never expires
  - Doesn’t have feature of whitelisting files based on file extension
  - Doesn’t have any protection within the core system that protects against a malicious attack via file upload
  - Doesn’t verify if the file is legitimate/ resulting data is valid
  - Every users have ability to use/ access all the features of the system

CODE ANALYSIS FINDINGS

SonarCloud



SWAMP



CODE REVIEW FINDINGS

SonarCloud

Blockers: (Not an Issue)

'PASSWORD' detected in this expression, review this potentially hard-coded credential.
Vulnerability Blocker Open Not assigned 30min effort Comment

CWE – 259, CWE-798

Refactor this code to not construct the path from tainted, user-controlled data. Why this issue? 11 years ago
Vulnerability Blocker Open Not assigned 30min effort Comment cwe, owasp-a5, sa

CWE's: 22, 23, 36, 99, 641

Critical:

Use Galois/Counter Mode (GCM/NoPadding) instead. Why this issue?
Vulnerability Critical Open Not assigned 2min effort Comment

CWE-327: Use of a Broken or Risky Cryptographic Algorithm

SWAMP

Finding Summary

Finding Type	Count
PMD / Best Practices / GuardLogStatement (page 4)	33
PMD / Performance / AvoidFileStream (page 15)	4
PMD / Error Prone / SingletonClassReturningNewInstance (page 16)	2
PMD / Code Style / OnlyOneReturn (page 18)	13
Error Handling (page 22)	8
PMD / Design / AvoidCatchingGenericException (page 25)	8
PMD / Code Style / FieldDeclarationsShouldBeAtStartOfClass (page 30)	7
PMD / Error Prone / AvoidLiteralsInIfCondition (page 27)	7
PMD / Performance / AvoidInstantiatingObjectsInLoops (page 33)	6
PMD / Design / CyclomaticComplexity (page 35)	5
PMD / Multithreading / UseConcurrentHashMap (page 36)	5
PMD / Error Prone / BeanMembersShouldSerialize (page 38)	4
PMD / Code Style / AtLeastOneConstructor (page 39)	3
PMD / Design / NPathComplexity (page 40)	3
PMD / Design / ExcessiveMethodLength (page 42)	2
PMD / Performance / RedundantFieldInitializer (page 41)	2
PMD / Performance / UseIndexOfChar (page 40)	2
PMD / Best Practices / UnusedFormalParameter (page 47)	1
PMD / Code Style / ConfusingTernary (page 44)	1
PMD / Design / AvoidRethrowingException (page 43)	1

HARD-CODED PASSWORD

'PASSWORD' detected in this expression, review this potentially hard-coded credential.

Vulnerability Blocker Open Not assigned 30min effort Comment

```
public static final String GP_PASSWORD_REQUIRES_NON_DIGIT = "security.passwordRequiresNonDigit";
```

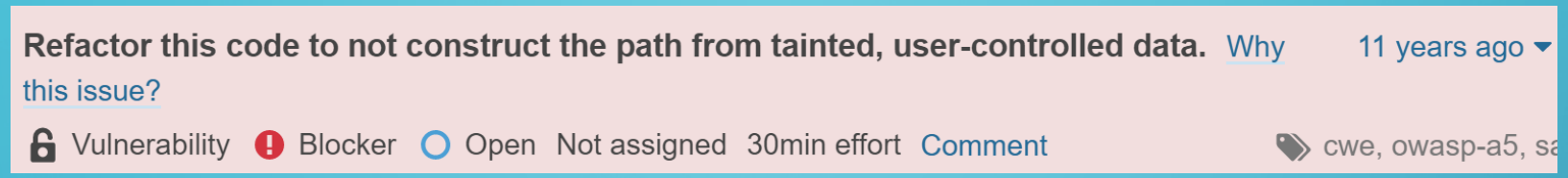
```
public static final String GP_PASSWORD_REQUIRES_UPPER_AND_LOWER_CASE = "security.passwordRequiresUpperAndLowerCase";
```

```
public static final String GP_PASSWORD_MINIMUM_LENGTH = "security.passwordMinimumLength";
```

Represented 14 out of 15 Blockers
Maps to the following CWEs

CWE-259: Use of Hard-Coded Password
CWE-798: Use of Hard-Coded Credentials

DIRECTORY TRAVERSAL IN FILE UPLOAD



CWE-22: Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)
CWE-23: Relative Path Traversal
CWE-36: Absolute Path Traversal
CWE-99: Improper Control of Resource Identifiers (Resource Injection)
CWE-641: Improper Restriction of Names for Files and Other Resources

`'../'`

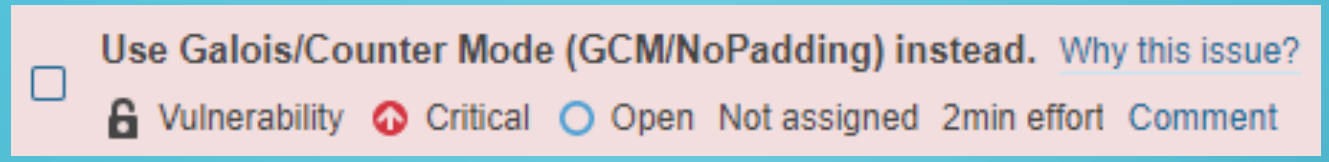
`"../bin"`

Solution: Sanitize input, whitelist allowed paths.

For Example:

```
// Restrict the username to letters and digits only
if (!user.matches("[a-zA-Z0-9]++")) {
  return false;
```

CRYPTOGRAPHIC ALGORITHM



- CWE-327: Use of a Broken or Risky Cryptographic Algorithm

From OpenMRS

```
 * Encryption properties; both vector and key are required to utilize a two-way encryption
 */
public static final String ENCRYPTION_CIPHER_CONFIGURATION = "AES/CBC/PKCS5Padding";
public static final String ENCRYPTION_KEY_SPEC = "AES";
public static final String ENCRYPTION_VECTOR_RUNTIME_PROPERTY = "encryption.vector";
public static final String ENCRYPTION_VECTOR_DEFAULT = "9wyBUNglFCRVSUhMfsTa3Q==";
public static final String ENCRYPTION_KEY_RUNTIME_PROPERTY = "encryption.key";
public static final String ENCRYPTION_KEY_DEFAULT = "dTfyELRrAICGDwzjHDjuhw==";
```

From SonarCloud

Cipher Block Chaining (CBC) with PKCS#5 padding (or PKCS#7) is susceptible to padding oracle attacks.

PROJECT CONTRIBUTION

We haven't made any contributions (yet) to the core project

We were security and vulnerability focused

- Complex part of the codebase

We didn't fully understand how these areas were implemented

- Difficult to comment on / suggest changes

CONTRIBUTION PROCESS WAS UNCLEAR

Very large project / codebase

- Split between core API, web interface, module library
- The openMRS core alone is 75k lines

There are 227 individual repositories that make up OpenMRS

- Most of which are current and deprecated modules

CONTRIBUTION PROCESS WAS UNCLEAR

* Github (Codebase)
* Confluence (Wiki)
* Jira (Issue tracking)
* Slack / IRC

ISSUES WORTH REPORTING

Hard coded credentials

- Default Username / Password for Scheduler Module

```
12  public class SchedulerConstants {
13
14          // Number of milliseconds per second (used for readability)
15          public static final int SCHEDULER_MILLIS_PER_SECOND = 1000;
16
17          // 0 second delay added before the initial start of a task
18          public static final long SCHEDULER_DEFAULT_DELAY = 0;
19
20          public static String SCHEDULER_DEFAULT_USERNAME = "admin";
21
22          public static String SCHEDULER_DEFAULT_PASSWORD = "test";
```

'PASSWORD' detected in this expression, review this potentially hard-coded credential. See Rule 2 months ago L22

Vulnerability Blocker Open Not assigned 30min effort cert, cwe, owasp-a2, sans-top25-porous

```
23
```

ISSUES WORTH REPORTING

Potential issues with crypto implementation

- Use of Galois Counter Mode
- Ensuring data can be encrypted

```
216         public static String encrypt(String text, byte[] initVector, byte[] secretKey) {
217                 IvParameterSpec initVectorSpec = new IvParameterSpec(initVector);
218                 SecretKeySpec secret = new SecretKeySpec(secretKey, OpenmrsConstants.ENCRYPTION_KEY_SPEC);
219                 byte[] encrypted;
220                 String result;
221
222                 try {
223                         Cipher cipher = Cipher.getInstance(OpenmrsConstants.ENCRYPTION_CIPHER_CONFIGURATION);
```

Use Galois/Counter Mode (GCM/NoPadding) instead. See Rule — 2 months ago ▾ L223

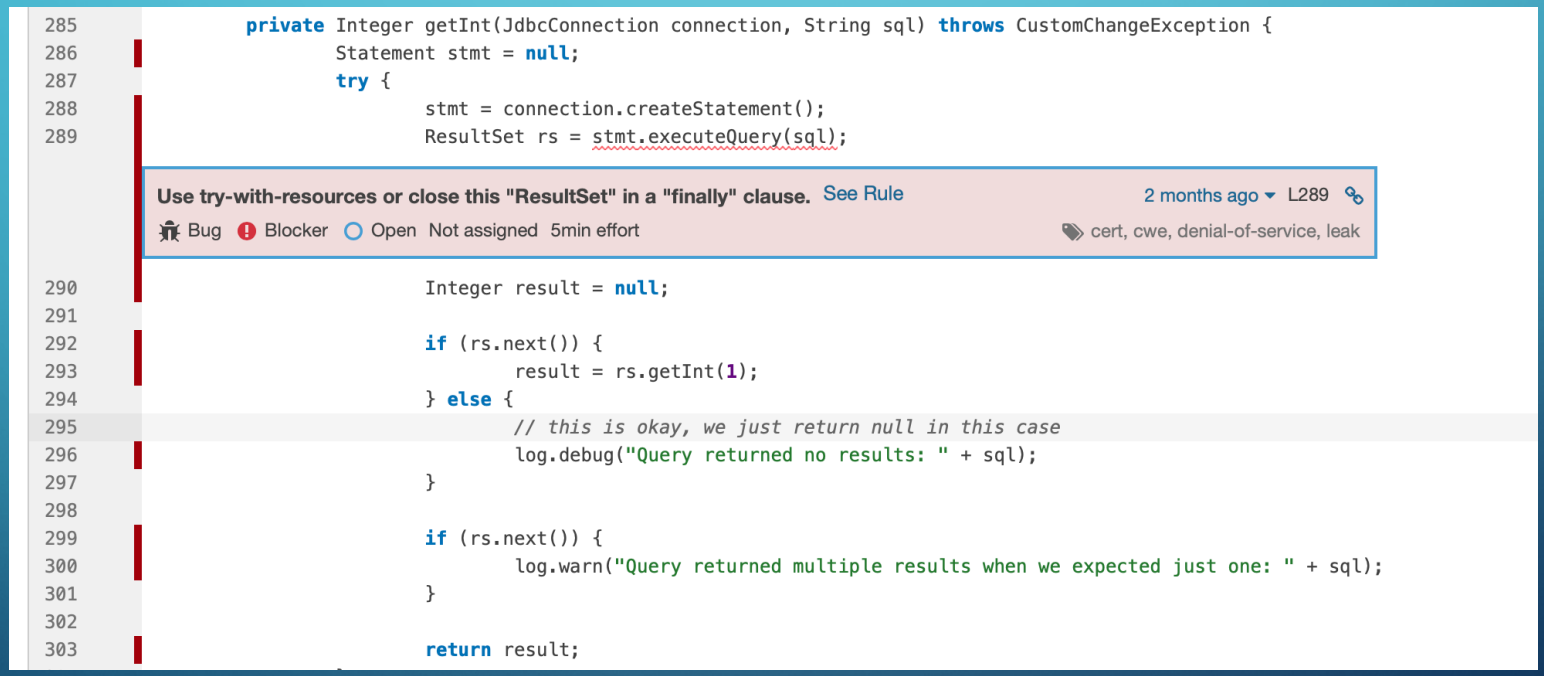
Vulnerability · Critical · Open · Not assigned · 2min effort — cert, cwe, owasp-a6, sans-top25-porous

```
224                         cipher.init(Cipher.ENCRYPT_MODE, secret, initVectorSpec);
225                         encrypted = cipher.doFinal(text.getBytes(StandardCharsets.UTF_8));
226                         result = new String(Base64.getEncoder().encode(encrypted), StandardCharsets.UTF_8);
227                 }
228                 catch (GeneralSecurityException e) {
229                         throw new APIException("could.not.encrypt.text", null, e);
230                 }
231
232                 return result;
233         }
234
```

ISSUES WORTH REPORTING

Closing resources in a `Finally` block to prevent resource leaks

- Use `AutoCloseable` class with "try-with-resources" pattern




```
285         private Integer getInt(JdbcConnection connection, String sql) throws CustomChangeException {
286             Statement stmt = null;
287             try {
288                 stmt = connection.createStatement();
289                 ResultSet rs = stmt.executeQuery(sql);
290                 Integer result = null;
291
292                 if (rs.next()) {
293                     result = rs.getInt(1);
294                 } else {
295                     // this is okay, we just return null in this case
296                     log.debug("Query returned no results: " + sql);
297                 }
298
299                 if (rs.next()) {
300                     log.warn("Query returned multiple results when we expected just one: " + sql);
301                 }
302
303                 return result;
```

ISSUES WORTH REPORTING

These issues correspond to the following CWEs

- CWE-798 - Use of Hard-coded Credentials
- CWE-327 - Use of Broken or Risky Cryptographic Algorithm
- CWE-459 - Incomplete Cleanup

SUMMARY / CONCLUSION

- Most assurance claims were found to be acceptable
- Without proper setup and administration, there are a lot of potential security issues
- Large community willing to help, but hard to know where to connect
- Lots of great documentation, but sometimes difficult to find current best practices
    - Result of this being a ~15 year old project

SOURCES AND LINKS

- OpenMRS Website: https://openmrs.org/
- OpenMRS demo: https://openmrs.org/demo/
- OpenMRS Github: https://github.com/openmrs/openmrs-core
- OpenMRS Wiki: https://wiki.openmrs.org/
- The Squadratics GitHub Repo: https://github.com/The-Squadratics/openMRS_security_project

QUESTIONS & ANSWERS

THANK YOU…!!