~~~~~~~~~~~~~~~ Summary ~~~~~~~~~~~~~~~
Scan Score - 69
Low - 21
Medium - 0
High - 0
Critical - 1

~~~~~~~~~~~~~~~ Vulnerability 1 ~~~~~~~~~~~~~~~
ID - CWE-111
Module - vulnerable.java (Line 4)
Name - Direct Use of Unsafe JNI
Description - When a Java application uses the Java Native Interface (JNI) to call code written in another programming language, it can expose the application to weaknesses in that code, even if those weaknesses cannot occur in Java.
Severity - Low
Resource - https://cwe.mitre.org/data/definitions/111.html
Remediation Advice - Implement error handling around JNI call; do not use JNI calls if native library is not trusted; use Java API equivalents if they exist
Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 2 ~~~~~~~~~~~~~~~
ID - CWE-500
Module - vulnerable.java (Line 7)
Name - Public Static Field Not Marked Final
Description - An object contains a public static field that is not marked final, which might allow it to be modified in unexpected ways.
Severity - Low
Resource - https://cwe.mitre.org/data/definitions/500.html
Remediation Advice - Clearly identify the scope for all critical data elements, including whether they should be regarded as static. Make any static fields private and constant.
Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 3 ~~~~~~~~~~~~~~~
ID - CWE-798
Module - vulnerable.java (Line 10)
Name - Use of Hard-coded Credentials
Description - The product contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data.
Severity - Low
Resource - https://cwe.mitre.org/data/definitions/798.html
Remediation Advice - Credentials should be hashed and stored safely in a password-protected external file
Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 4 ~~~~~~~~~~~~~~~

ID - CWE-259

Module - vulnerable.java (Line 14)

Name - Use of Hard-coded Password

Description - The product contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/259.html

Remediation Advice - Passwords should be hashed and stored safely in a password-protected external file

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 5 ~~~~~~~~~~~~~~~

ID - CWE-321

Module - vulnerable.java (Line 17)

Name - Use of Hard-coded Cryptographic Key

Description - The use of a hard-coded cryptographic key significantly increases the possibility that encrypted data may be recovered.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/321.html

Remediation Advice - Cryptographic keys should be stored safely in a password-protected external file

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 6 ~~~~~~~~~~~~~~~

ID - CWE-397

Module - vulnerable.java (Line 20)

Name - Declaration of Throws for Generic Exception

Description - Throwing overly broad exceptions promotes complex error handling code that is more likely to contain security vulnerabilities.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/397.html

Remediation Advice - Define the specific exceptions that should be thrown.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 7 ~~~~~~~~~~~~~~~

ID - CWE-481

Module - vulnerable.java (Line 26)

Name - Assigning instead of Comparing

Description - The code uses an operator for assignment when the intention was to perform a comparison.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/481.html

Remediation Advice - Check operator used is correct. For example == is used for comparison and = is used for assignment.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 8 ~~~~~~~~~~~~~~~

ID - CWE-491

Module - vulnerable.java (Line 32)

Name - Public cloneable() Method Without Final ('Object Hijack')

Description - A class has a cloneable() method that is not declared final, which allows an object to be created without calling the constructor. This can cause the object to be in an unexpected state.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/491.html

Remediation Advice - Make the cloneable() method final.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 9 ~~~~~~~~~~~~~~~

ID - CWE-493

Module - vulnerable.java (Line 37)

Name - Critical Public Variable Without Final Modifier

Description - The product has a critical public variable that is not final, which allows the variable to be modified to contain unexpected values.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/493.html

Remediation Advice - Declare all public fields as final when possible, especially if it is used to maintain internal state of an Applet or of classes used by an Applet. If a field must be public, then perform all appropriate sanity checks before accessing the field from your code.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 10 ~~~~~~~~~~~~~~~

ID - CWE-582

Module - vulnerable.java (Line 40)

Name - Array Declared Public, Final, and Static

Description - The product declares an array public, final, and static, which is not sufficient to prevent the array's contents from being modified.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/582.html

Remediation Advice - The array should be made private.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 11 ~~~~~~~~~~~~~~~

ID - CWE-583

Module - vulnerable.java (Line 43)

Name - finalize() Method Declared Public

Description - The product violates secure coding principles for mobile code by declaring a finalize() method public.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/583.html

Remediation Advice - If you are using finalize() as it was designed, there is no reason to declare finalize() with anything other than protected access.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 12 ~~~~~~~~~~~~~~~

ID - CWE-595

Module - vulnerable.java (Line 49)

Name - Comparison of Object References Instead of Object Contents

Description - The product compares object references instead of the contents of the objects themselves, preventing it from detecting equivalent objects

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/595.html

Remediation Advice - The equals() method should be used used to compare objects instead of ==

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 13 ~~~~~~~~~~~~~~~

ID - CWE-585

Module - vulnerable.java (Line 56)

Name - Empty Synchronized Block

Description - An empty synchronized block does not actually accomplish any synchronization and may indicate a troubled section of code. An empty synchronized block can occur because code no longer needed within the synchronized block is commented out without removing the synchronized block.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/585.html

Remediation Advice - Remove empty synchronised block or define procedures that access or modify data that is exposed to multiple threads

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 14 ~~~~~~~~~~~~~~~

ID - CWE-89

Module - vulnerable.java (Line 68)

Name - Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Description - The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component

Severity - Critical

Resource - https://cwe.mitre.org/data/definitions/89.html

Remediation Advice - Prepared statements, client and server side input validation, safe stored procedures, or escaping user input can be used to mitigate against SQL injection attacks.

Days To Remediate - 15

~~~~~~~~~~~~~~~ Vulnerability 15 ~~~~~~~~~~~~~~~

ID - CWE-209

Module - vulnerable.java (Line 72)

Name - Generation of Error Message Containing Sensitive Information

Description - The product generates an error message that includes sensitive information about its environment, users, or associated data.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/209.html

Remediation Advice - When an exception is caught, only print insensitive and desired data to a user.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 16 ~~~~~~~~~~~~~~~

ID - CWE-246

Module - vulnerable.java (Line 81)

Name - J2EE Bad Practices: Direct Use of Sockets

Description - The J2EE application directly uses sockets instead of using framework method calls.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/246.html

Remediation Advice - Use framework method calls instead of using sockets directly.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 17 ~~~~~~~~~~~~~~~

ID - CWE-326

Module - vulnerable.java (Line 85)

Name - Inadequate Encryption Strength

Description - The product stores or transmits sensitive data using an encryption scheme that is theoretically sound, but is not strong enough for the level of protection required.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/326.html

Remediation Advice - Use trusted libraries/APIs that create encryption keys using best practices or define long, complex strings with a variety of letters, digits, upper and lowercase, and special characters.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 18 ~~~~~~~~~~~~~~~

ID - CWE-382

Module - vulnerable.java (Line 94)

Name - J2EE Bad Practices: Use of System.exit()

Description - A J2EE application uses System.exit(), which also shuts down its container.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/382.html

Remediation Advice - Shutdown function should be a privileged function only available to an authorised administrative user. Web applications should not call System.exit() as it can cause the virtual machine to exit.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 19 ~~~~~~~~~~~~~~~

ID - CWE-395

Module - vulnerable.java (Line 100)

Name - Use of NullPointerException Catch to Detect NULL Pointer Dereference

Description - Catching NullPointerException should not be used as an alternative to programmatic checks to prevent de-referencing a null pointer.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/395.html

Remediation Advice - Do not extensively rely on catching exceptions (especially for validating user input) to handle errors. Handling exceptions can decrease the performance of an application.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 20 ~~~~~~~~~~~~~~~

ID - CWE-396

Module - vulnerable.java (Line 107)

Name - Declaration of Catch for Generic Exception

Description - Catching overly broad exceptions promotes complex error handling code that is more likely to contain security vulnerabilities.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/396.html

Remediation Advice - Define specific exceptions and use multiple catch blocks if necessary.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 21 ~~~~~~~~~~~~~~~

ID - CWE-572

Module - vulnerable.java (Line 113)

Name - Call to Thread run() instead of start()

Description - The product calls a thread's run() method instead of calling start(), which causes the code to run in the thread of the caller instead of the callee.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/572.html

Remediation Advice - Use the start() method instead of the run() method.

Days To Remediate - 120

~~~~~~~~~~~~~~~ Vulnerability 22 ~~~~~~~~~~~~~~~

ID - CWE-586

Module - vulnerable.java (Line 116)

Name - Explicit Call to Finalize()

Description - The product makes an explicit call to the finalize() method from outside the finalizer.

Severity - Low

Resource - https://cwe.mitre.org/data/definitions/586.html

Remediation Advice - Do not make explicit calls to finalize().

Days To Remediate - 120