STR51-J. Use the charset encoder and decoder classes when more control over the encoding process is required

String objects in Java are encoded in UTF-16. Java Platform is required to support other character encodings or charsets such as US-ASCII, ISO-8859-1, and UTF-8. Errors may occur when converting between differently coded character data. There are two general types of encoding errors. If the byte sequence is not valid for the specified charset then the input is considered malformed. If the byte sequence cannot be mapped to an equivalent character sequence then an unmappable character has been encountered.

According to the Java API [API 2014] for the String constructors:

```
The behavior of this constructor when the given bytes are not valid in the given charset is unspecified.
```

Similarly, the description of the String.getBytes(Charset) method states:

```
This method always replaces malformed-input and unmappable-character sequences with this charset's default replacement byte array.
```

The CharsetEncoder class is used to transform character data into a sequence of bytes in a specific charset. The input character sequence is provided in a character buffer or a series of such buffers. The output byte sequence is written to a byte buffer or a series of such buffers. The CharsetDecoder class reverses this process by transforming a sequence of bytes in a specific charset into character data. The input byte sequence is provided in a byte buffer or a series of such buffers, while the output character sequence is written to a character buffer or a series of such buffers.

Special care should be taken when decoding untrusted byte data to ensure that malformed input or unmappable character errors do not result in defects and vulnerabilities. Encoding errors can also occur, for example, encoding a cryptographic key containing malformed input for transmission will result in an error. Encoding and decoding errors typically result in data corruption.

Noncompliant Code Example

This noncompliant code example is similar to the one used in STR03-J. Do not represent numeric data as strings in that it attempts to convert a byte array containing the two's-complement representation of this BigInteger value to a String. Because the byte array contains malformed-input sequences, the behavior of the String constructor is unspecified.

```java
import java.math.BigInteger;
import java.nio.CharBuffer;

public class CharsetConversion {
    public static void main(String[] args) {
        BigInteger x = new BigInteger("530500452766");
        byte[] byteArray = x.toByteArray();
        String s = new String(byteArray);
        System.out.println(s);
    }
}
```

Compliant Solution

The java.nio.charset.CharsetEncoder and java.nio.charset.CharacterDecoder provide greater control over the process. In this compliant solution, the CharsetDecoder.decode() method is used to convert the byte array containing the two's-complement representation of this BigInteger value to a CharBuffer. Because the bytes do not represent a valid UTF-16, the input is considered malformed and a MalformedURLException is thrown.
import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
import java.nio.charset.CharacterCodingException;
import java.nio.charset.CharsetDecoder;
import java.nio.charset.MalformedInputException;
import java.nio.charset.StandardCharsets;
import java.nio.charset.UnmappableCharacterException;

public class CharsetConversion {
    public static void main(String[] args) {
        CharBuffer charBuffer;
        CharsetDecoder decoder = StandardCharsets.UTF_16.newDecoder();
        BigInteger x = new BigInteger("530500452766");
        byte[] byteArray = x.toByteArray();
        ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray);
        try {
            charBuffer = decoder.decode(byteBuffer);
            s = charBuffer.toString();
            System.out.println(s);
        } catch (IllegalStateException e) {
            e.printStackTrace();
        } catch (MalformedInputException e) {
            e.printStackTrace();
        } catch (UnmappableCharacterException e) {
            e.printStackTrace();
        } catch (CharacterCodingException e) {
            e.printStackTrace();
        }
    }
}

Risk Assessment

Malformed input or unmappable character errors can result in a loss of data integrity.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR05-J</td>
<td>low</td>
<td>unlikely</td>
<td>medium</td>
<td>P2</td>
<td>L3</td>
</tr>
</tbody>
</table>

Related Guidelines

- MITRE CWE
  - CWE-838. Inappropriate Encoding for Output Context
  - CWE-116. Improper Encoding or Escaping of Output

Bibliography

[API 2006] Class String