EXP35-PL. Use the correct operator type for comparing values

Perl provides two sets of comparison operators: one set for working with numbers and one set for working with strings.

<table>
<thead>
<tr>
<th>Numbers</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>eq</td>
</tr>
<tr>
<td>!=</td>
<td>ne</td>
</tr>
<tr>
<td>&lt;</td>
<td>lt</td>
</tr>
<tr>
<td>&lt;=</td>
<td>le</td>
</tr>
<tr>
<td>&gt;</td>
<td>gt</td>
</tr>
<tr>
<td>&gt;=</td>
<td>ge</td>
</tr>
<tr>
<td>&lt;=&gt;</td>
<td>cmp</td>
</tr>
</tbody>
</table>

Do not use the number comparison operators on nonnumeric strings. Likewise, do not use the string comparison operators on numbers.

Noncompliant Code Example (Numbers)

This noncompliant code example improperly uses eq to test two numbers for equality.

```perl
my $num = 2;
print "Enter a number\n";
my $user_num = <STDIN>;
chomp $user_num;
if ($num eq $user_num) {print "true\n"} else {print "false\n"};
```

This code will print true if the user enters 2, but false if the user enters 02.

Compliant Solution (Numbers)

This compliant solution uses ==, which interprets its arguments as numbers. This code therefore prints true even if the right argument to == is initialized to some different string like 02.

```perl
my $num = 2;
print "Enter a number\n";
my $user_num = <STDIN>;
chomp $user_num;
if ($num == $user_num) {print "true\n"} else {print "false\n"};
```

Noncompliant Code Example (Strings)

This noncompliant code example improperly uses == to test two strings for equality.

```perl
sub check_password {  
    my $correct = shift;
    my $password = shift;
    # encrypt password
    if ($password == $correct) {  
        return true;
    } else {  
        return false;
    }
}
```
The `==` operator first converts its arguments into numbers by extracting digits from the front of each argument (along with a preceding + or -). Nonnumeric data in an argument is ignored, and the number consists of whatever digits were extracted. A string such as "goodpass" has no leading digits, so it is converted to the numeral 0. Consequently, unless either $password or $correct contains leading digits, they will both be converted to 0 and will be considered equivalent.

Compliant Solution (Strings)

This compliant solution uses `eq`, which interprets its arguments as strings.

```perl
sub check_password {
    my $correct = shift;
    my $password = shift;
    # encrypt password
    if ($password eq $correct) {
        return true;
    } else {
        return false;
    }
}
```

Risk Assessment

Confusing the string comparison operators with numeric comparison operators can lead to incorrect program behavior or incorrect program data.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP35-PL</td>
<td>Low</td>
<td>Likely</td>
<td>Low</td>
<td>P9</td>
<td>L2</td>
</tr>
</tbody>
</table>

Automated Detection

<table>
<thead>
<tr>
<th>Tool</th>
<th>Diagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perl::Critic</td>
<td>ValuesAndExpressions::ProhibitMismatchedOperators</td>
</tr>
</tbody>
</table>

Bibliography

- [CPAN] Elliot Shank, Perl-Critic-1.116 ProhibitMismatchedOperators
- [Wall 2011] perlmod