MSC01-PL. Detect and remove unused variables

The presence of unused variables may indicate significant logic errors. To prevent such errors, unused values should be identified and removed from code.

Noncompliant Code Example

This noncompliant code example contains a variable $new_name that is initialized but never subsequently read.

```perl
sub fix_name {
    my $name = shift;
    my $new_name = $name;
    $name =~ s/^([a-z])$/\U$1\E/g;
    $name =~ s/ ([a-z])/ \U$1\E/g;
    return $name;
}
```

Compliant Solution

This compliant solution eliminates the unused variable

```perl
sub fix_name {
    my $name = shift;
    $name =~ s/^([a-z])$/\U$1\E/g;
    $name =~ s/ ([a-z])/ \U$1\E/g;
    return $name;
}
```

Risk Assessment

The presence of unused variables may indicate logic errors that can lead to unintended program behavior. As a result, resolving unused variables can be an in-depth process requiring significant analysis.

<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>MSC01-PL</td>
<td>Low</td>
<td>Unlikely</td>
<td>High</td>
<td>P1</td>
<td>L3</td>
</tr>
</tbody>
</table>

Automated Detection

<table>
<thead>
<tr>
<th>Tool</th>
<th>Diagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perl::Critic</td>
<td>Variables::ProhibitUnusedVariables</td>
</tr>
</tbody>
</table>

Related Guidelines

- SEI CERT C Coding Standard: MSC13-C. Detect and remove unused values
- SEI CERT C++ Coding Standard: VOID MSC13-CPP. Detect and remove unused values

Bibliography

- [CPAN] Elliot Shank, Perl-Critic-1.116 Variables::ProhibitUnusedVariables