EXP34-PL. Do not modify \$_ in list or sorting functions

Perl provides several functions for list manipulation. For instance, the `map()` function takes an expression or block, applies it to each element in a list, and returns the list of mapped elements. If it is given a block, the block is executed with \$_ assigned to each element of the list in turn. The `perlfun` manpage adds:

```
Note that \$_ is an alias to the list value, so it can be used to modify the elements of the LIST. While this is useful and supported, it can cause bizarre results if the elements of LIST are not variables. Using a regular "foreach" loop for this purpose would be clearer in most cases.
```

Although it is supported, using `map()` to modify a list in place can lead to surprises in maintainability and is thus forbidden.

Many other list functions provide similar functionality, using a block on various list elements. The `grep()` function is one such example, as are the `first()` and `reduce()` functions in `List::Util` and all of the functions in `List::MoreUtils`.

Finally, the `sort()` function also provides aliases to its comparison blocks, so a comparison block for `sort()` must not modify its variables.

**Noncompliant Code Example (grep())**

This noncompliant code example reads the `/etc/passwd` file and lists all users who use `/bin/sh` as their login shell.

```perl
open( PASSWD, "<", "/etc/passwd") or croak "error opening /etc/passwd: stopped"
my @users = <PASSWD>;
my @shell_users = grep +(s|/bin/sh||), @users;
foreach my $user (@shell_users) {
    print "Shell User: $user";
}
```

However, because the `grep()` block removes `/bin/sh` from any input line that contains it, it modifies the `@users` list so that no user has `/bin/sh`!

**Compliant Solution (grep())**

This compliant solution does the same thing but does not modify the `@users` array.

```perl
open( PASSWD, "<", "/etc/passwd") or croak "error opening /etc/passwd: stopped"
my @users = <PASSWD>;
my @shell_users = grep +(m|/bin/sh|), @users;
foreach my $user (@shell_users) {
    $user =~ s|/bin/sh||;
    print "Shell User: $user";
}
```

**Compliant Solution (apply())**

This compliant solution does the same thing but uses `List::MoreUtils::apply()`, which guarantees not to modify its input list.

```perl
open( PASSWD, "<", "/etc/passwd") or croak "error opening /etc/passwd: stopped"
my @users = <PASSWD>;
my @shell_users = List::MoreUtils::apply { s|/bin/sh|| } @users;
foreach my $user (@shell_users) {
    print "Shell User: $user";
}
```

**Risk Assessment**

Failure to handle error codes or other values returned by functions can lead to incorrect program flow and violations of data integrity.

<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>EXP34-PL</td>
<td>Medium</td>
<td>Likely</td>
<td>Low</td>
<td>P18</td>
<td>L1</td>
</tr>
</tbody>
</table>
Automated Detection

<table>
<thead>
<tr>
<th>Tool</th>
<th>Diagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perl::Critic</td>
<td>ControlStructures::ProhibitMutatingListFunctions</td>
</tr>
</tbody>
</table>

Bibliography

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CPAN] Bar, Graham</td>
<td>List::Utils</td>
</tr>
<tr>
<td>[CPAN] Kennedy, Adam</td>
<td>List::MoreUtils</td>
</tr>
<tr>
<td>[Wall 2011]</td>
<td>perlfunc</td>
</tr>
</tbody>
</table>