API08-C. Avoid parameter names in a function prototype

Naming the parameters in a prototype declaration should never be necessary, and is often unwise, because these names can be affected by macro definitions.

Although the scope of an identifier in a function prototype begins at its declaration and ends at the end of that function's declarator, this scope is ignored by the preprocessor. Consequently, an identifier in a prototype having the same name as that of an existing macro is treated as an invocation of that macro.

Safeguarding parameter names is particularly critical in standard and system headers where the user expects to be able to include the header and have only the function names visible.

Noncompliant Code Example

This noncompliant code example,

```c
#define status 23
void update_status(int status);
```

generates an error, because the prototype, after preprocessing, becomes

```c
void update_status(int 23);
```

Perhaps more surprising is what happens if status is defined:

```c
#define status []
```

Then the resulting prototype is

```c
void update_status(int []);
```

which is syntactically correct but semantically quite different from the intent.

Compliant Solution

To protect an API's header prototypes from such misinterpretation, the developer must write them to avoid these surprises. Possible solutions include not using identifiers in prototypes, as in this example:

```c
void update_status(int);
```

Another solution is to comment them out, as in this example:

```c
void update_status(int /* status */);
```

Comments are converted to a single whitespace character in translation phase two.

Risk Assessment

Failure to protect header prototypes from misinterpretation can result in type errors in the program.

<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>API08-C</td>
<td>Medium</td>
<td>Unlikely</td>
<td>Medium</td>
<td>P4</td>
<td>L3</td>
</tr>
</tbody>
</table>

Automated Detection

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Checker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astére</td>
<td>19.04</td>
<td>named-declaration-parameter</td>
<td>Fully checked</td>
</tr>
<tr>
<td>ECLAIR</td>
<td>1.2</td>
<td>CC2.API08</td>
<td>Fully implemented</td>
</tr>
</tbody>
</table>
Related Vulnerabilities

Search for vulnerabilities resulting from the violation of this rule on the CERT website.

Bibliography

[C99 Rationale 2003]