STR04-C. Use plain char for characters in the basic character set

There are three character types: char, signed char, and unsigned char. Compilers have the latitude to define char to have the same range, representation, and behavior as either signed char or unsigned char. Irrespective of the choice made, char is a separate type from the other two and is not compatible with either.

For characters in the basic character set, it does not matter which data type is used, except for type compatibility. Consequently, it is best to use plain char for character data for compatibility with standard string-handling functions.

In most cases, the only portable operators on plain char types are assignment and equality operators (=, ==, !=). An exception is the translation to and from digits. For example, if the char c is a digit, c - '0' is a value between 0 and 9.

Noncompliant Code Example

This noncompliant code example simply shows the standard string-handling function strlen() being called with a plain character string, a signed character string, and an unsigned character string. The strlen() function takes a single argument of type const char *:

```c
size_t len;
char cstr[] = "char string";
signed char scstr[] = "signed char string";
unsigned char ucstr[] = "unsigned char string"

len = strlen(cstr);
len = strlen(scstr); /* Warns when char is unsigned */
len = strlen(ucstr); /* Warns when char is signed */
```

Compiling at high warning levels in compliance with MSC00-C. Compile cleanly at high warning levels causes warnings to be issued when

- Converting from unsigned char[] to const char * when char is signed
- Converting from signed char[] to const char * when char is defined to be unsigned

Casts are required to eliminate these warnings, but excessive casts can make code difficult to read and hide legitimate warning messages.

If this C code were compiled using a C++ compiler, conversions from unsigned char[] to const char * and from signed char[] to const char * would be flagged as errors requiring casts.

Compliant Solution

The compliant solution uses plain char for character data:

```c
size_t len;
char cstr[] = "char string";
len = strlen(cstr);
```

Conversions are not required, and the code compiles cleanly at high warning levels without casts.

Risk Assessment

Failing to use plain char for characters in the basic character set can lead to excessive casts and less effective compiler diagnostics.

<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>STR04-C</td>
<td>Low</td>
<td>Unlikely</td>
<td>Low</td>
<td>P3</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>Axivion Bauhaus Suite</td>
<td>6.9.0</td>
<td>CertC-STR04</td>
<td></td>
</tr>
</tbody>
</table>
### Related Vulnerabilities

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

### Related Guidelines

<table>
<thead>
<tr>
<th>SEI CERT C++ Coding Standard</th>
<th>VOID STR04-CPP. Use plain char for characters in the basic character set</th>
</tr>
</thead>
</table>
| MISRA C:2012                | Rule 10.1 (required)  
  Rule 10.2 (required)  
  Rule 10.3 (required)  
  Rule 10.4 (required) |