STR38-C. Do not confuse narrow and wide character strings and functions

Passing narrow string arguments to wide string functions or wide string arguments to narrow string functions can lead to unexpected and undefined behavior. Scaling problems are likely because of the difference in size between wide and narrow characters. (See ARR39-C. Do not add or subtract a scaled integer to a pointer.) Because wide strings are terminated by a null wide character and can contain null bytes, determining the length is also problematic.

Because `wchar_t` and `char` are distinct types, many compilers will produce a warning diagnostic if an inappropriate function is used. (See MSC00-C. Compile cleanly at high warning levels.)

Noncompliant Code Example (Wide Strings with Narrow String Functions)

This noncompliant code example incorrectly uses the `strncpy()` function in an attempt to copy up to 10 wide characters. However, because wide characters can contain null bytes, the copy operation may end earlier than anticipated, resulting in the truncation of the wide string.

```
#include <stddef.h>
#include <string.h>

void func(void) {
  wchar_t wide_str1[]  = L"0123456789";
  wchar_t wide_str2[] =  L"0000000000";

  strncpy(wide_str2, wide_str1, 10);
}
```

Noncompliant Code Example (Narrow Strings with Wide String Functions)

This noncompliant code example incorrectly invokes the `wcsncpy()` function to copy up to 10 wide characters from `narrow_str1` to `narrow_str2`. Bec ause `narrow_str2` is a narrow string, it has insufficient memory to store the result of the copy and the copy will result in a buffer overflow.

```
#include <wchar.h>

void func(void) {
  char narrow_str1[] = "01234567890123456789";
  char narrow_str2[] = "0000000000";

  wcsncpy(narrow_str2, narrow_str1, 10);
}
```

Compliant Solution

This compliant solution uses the proper-width functions. Using `wcsncpy()` for wide character strings and `strncpy()` for narrow character strings ensures that data is not truncated and buffer overflow does not occur.

```
#include <string.h>
#include <wchar.h>

void func(void) {
  wchar_t wide_str1[] = L"0123456789";
  wchar_t wide_str2[] = L"0000000000";
  /* Use of proper-width function */
  wcsncpy(wide_str2, wide_str1, 10);

  char narrow_str1[] = "0123456789";
  char narrow_str2[] = "0000000000";
  /* Use of proper-width function */
  strncpy(narrow_str2, narrow_str1, 10);
}
```

Noncompliant Code Example (`strlen()`)

In this noncompliant code example, the `strlen()` function is used to determine the size of a wide character string:

```
#include <stdlib.h>
#include <string.h>

void func(void) {
  wchar_t wide_str1[] = L"0123456789";
  wchar_t *wide_str2 = (wchar_t*)malloc(strlen(wide_str1) + 1);
  if (wide_str2 == NULL) {
    /* Handle error */
  }
  /* ... */
  free(wide_str2);
  wide_str2 = NULL;
}
```

The `strlen()` function determines the number of characters that precede the terminating null character. However, wide characters can contain null bytes, particularly when expressing characters from the ASCII character set, as in this example. As a result, the `strlen()` function will return the number of bytes preceding the first null byte in the wide string.

Compliant Solution

This compliant solution correctly calculates the number of bytes required to contain a copy of the wide string, including the terminating null wide character:

```
#include <stdlib.h>
#include <wchar.h>

void func(void) {
  wchar_t wide_str1[] = L"0123456789";
  wchar_t *wide_str2 = (wchar_t *)malloc(
    (wcslen(wide_str1) + 1) * sizeof(wchar_t));
  if (wide_str2 == NULL) {
    /* Handle error */
  }
  /* ... */

  free(wide_str2);
  wide_str2 = NULL;
}
```

Risk Assessment

Confusing narrow and wide character strings can result in buffer overflows, data truncation, and other defects.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
STR38-C	High	Likely	Low	P27	L1

Automated Detection

Modern compilers recognize the difference between a `char *` and a `wchar_t *`, so compiling code that violates this rule will generate warnings. It is feasible to have automated software that recognizes functions of improper width and replaces them with functions of proper width (that is, software that uses `wcsncpy()` when it recognizes that the parameters are of type `wchar_t *`).

Tool	Version	Checker	Description
Astrée	19.04	**wide-narrow-string-cast** **wide-narrow-string-cast-implicit**	Partially checked
Axivion Bauhaus Suite	6.9.0	**CertC-STR38**	Fully implemented
Clang	3.9	`-Wincompatible-pointer-types`	
Coverity	2017.07	**PW**	Implemented
Parasoft C/C++test	10.4.2	**CERT_C-STR38-a**	Do not confuse narrow and wide character strings and functions
Polyspace Bug Finder	R2019b	CERT C: Rule STR38-C	Checks for misuse of narrow or wide character string (rule fully covered)

PRQA QA-C	9.5	**0432**	
PRQA QA-C++	4.3	**0403**	
RuleChecker	19.04	**wide-narrow-string-cast** **wide-narrow-string-cast-implicit**	Partially checked
TrustInSoft Analyzer	1.38	**pointer arithmetic**	Partially verified.

Related Vulnerabilities

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

Bibliography

[ISO/IEC 9899:2011]	7.24.2.4, "The `strncpy` Function" 7.29.4.2.2, "The `wcsncpy` Function"