STR03-C. Do not inadvertently truncate a string

Alternative functions that limit the number of bytes copied are often recommended to mitigate buffer overflow vulnerabilities. Examples include

- `strncpy()` instead of `strcpy()`
- `strncat()` instead of `strcat()`
- `fgets()` instead of `gets()`
- `snprintf()` instead of `sprintf()`

These functions truncate strings that exceed the specified limits. Additionally, some functions, such as `strncpy()`, do not guarantee that the resulting character sequence is null-terminated. (See STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string.)

Unintentional truncation results in a loss of data and in some cases leads to software vulnerabilities.

Noncompliant Code Example

The standard functions `strncpy()` and `strncat()` copy a specified number of characters `n` from a source string to a destination array. In the case of `strncpy()`, if there is no null character in the first `n` characters of the source array, the result will not be null-terminated and any remaining characters are truncated.

```
char *string_data;
char a[16];
/* ... */
strncpy(a, string_data, sizeof(a));
```

Compliant Solution (Adequate Space)

Either the `strcpy()` or `strncpy()` function can be used to copy a string and a null character to a destination buffer, provided there is enough space. The programmer must be careful to ensure that the destination buffer is large enough to hold the string to be copied and the null byte to prevent errors, such as data truncation and buffer overflow.

```
char *string_data = NULL;
char a[16];

/* ... */

if (string_data == NULL) {
  /* Handle null pointer error */
}
else if (strlen(string_data) >= sizeof(a)) {
  /* Handle overlong string error */
}
else {
  strcpy(a, string_data);
}
```

This solution requires that `string_data` is null-terminated; that is, a null byte can be found within the bounds of the referenced character array. Otherwise, `strlen()` will stray into other objects before finding a null byte.

Compliant Solution (`strcpy_s()`, C11 Annex K)

The `strcpy_s()` function defined in C11 Annex K [ISO/IEC 9899:2011] provides additional safeguards, including accepting the size of the destination buffer as an additional argument. (See STR07-C. Use the bounds-checking interfaces for string manipulation.) Also, `strnlen_s()` accepts a maximum-length argument for strings that may not be null-terminated.

```
char *string_data = NULL;
char a[16];

/* ... */

if (string_data == NULL) {
  /* Handle null pointer error */
}
else if (strnlen_s(string_data, sizeof(a)) >= sizeof(a)) {
  /* Handle overlong string error */
}
else {
  strcpy_s(a, sizeof(a), string_data);
}
```

If a runtime-constraint error is detected by the call to either `strnlen_s()` or `strcpy_s()`, the currently registered runtime-constraint handler is invoked. See ERR03-C. Use runtime-constraint handlers when calling the bounds-checking interfaces for more information on using runtime-constraint handlers with C11 Annex K functions.

Exceptions

STR03-C-EX1: The intent of the programmer is to purposely truncate the string.

Risk Assessment

Truncating strings can lead to a loss of data.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
STR03-C	Medium	Probable	Medium	**P8**	**L2**

Automated Detection

Tool	Version	Checker	Description
CodeSonar	5.2p0	**MISC.MEM.NTERM**	No Space For Null Terminator
Compass/ROSE			Could detect violations in the following manner: all calls to `strncpy()` and the other functions should be followed by an assignment of a terminating character to null-terminate the string
GCC	8.1	`-Wstringop-truncation`	Detects string truncation by `strncat` and `strncpy`.
Klocwork	2018	**NNTS.MIGHT NNTS.MUST**	
LDRA tool suite	9.7.1	**115 S, 44 S**	Partially implemented
Parasoft C/C++test	10.4.2	**CERT_C-STR03-a**	Avoid overflow due to reading a not zero terminated string
Polyspace Bug Finder	R2019b	CERT C: Rec. STR03-C	Checks for invalid use of standard library string routine (rec. partially supported)

Related Vulnerabilities

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

Related Guidelines

SEI CERT C++ Coding Standard	VOID STR03-CPP. Do not inadvertently truncate a null-terminated character array
ISO/IEC TR 24772:2013	String Termination [CJM]
MITRE CWE	CWE-170, Improper null termination CWE-464, Addition of data structure sentinel

Bibliography

[Seacord 2013]	Chapter 2, "Strings"