POS30-C. Use the readlink() function properly

The readlink() function reads where a link points to. It makes no effort to null-terminate its second argument, buffer. Instead, it just returns the number of characters it has written.

Noncompliant Code Example

If len is equal to sizeof(buf), the null terminator is written 1 byte past the end of buf:

```c
char buf[1024];
ssize_t len = readlink("/usr/bin/perl", buf, sizeof(buf));
buf[len] = '\0';
```

An incorrect solution to this problem is to try to make buf large enough that it can always hold the result:

```c
long symlink_max;
size_t bufsize;
char *buf;
ssize_t len;
errno = 0;
symlink_max = pathconf("/usr/bin/", _PC_SYMLINK_MAX);
if (symlink_max == -1) {
    if (errno != 0) {
        /* handle error condition */
    }
    bufsize = 10000;
} else {
    bufsize = symlink_max+1;
}
buf = (char *)malloc(bufsize);
if (buf == NULL) {
    /* handle error condition */
}
len = readlink("/usr/bin/perl", buf, bufsize);
buf[len] = '\0';
```

This modification incorrectly assumes that the symbolic link cannot be longer than the value of SYMLINK_MAX returned by pathconf(). However, the value returned by pathconf() is out of date by the time readlink() is called, so the off-by-one buffer-overflow risk is still present because, between the two calls, the location of /usr/bin/perl can change to a file system with a larger SYMLINK_MAX value. Also, if SYMLINK_MAX is indeterminate (that is, if pathconf() returned -1 without setting errno), the code uses an arbitrary large buffer size (10,000) that it hopes will be sufficient, but there is a small chance that readlink() can return exactly this size.

An additional issue is that readlink() can return -1 if it fails, causing an off-by-one underflow.

Compliant Solution

This compliant solution ensures there is no overflow by reading in only sizeof(buf)-1 characters. It also properly checks to see if an error has occurred:

```c
enum { BUFFERSIZE = 1024 }; char buf[BUFFERSIZE];
ssize_t len = readlink("/usr/bin/perl", buf, sizeof(buf)-1);
if (len != -1) {
    buf[len] = '\0';
} else {
    /* handle error condition */
}
```

Risk Assessment
Failing to properly null-terminate the result of `readlink()` can result in abnormal program termination and buffer-overflow vulnerabilities.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediaion Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS30-C</td>
<td>high</td>
<td>probable</td>
<td>medium</td>
<td>P12</td>
<td>L1</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-POS30</td>
<td></td>
</tr>
<tr>
<td>Compass/ROSE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverity</td>
<td>2017.07</td>
<td>READLINK</td>
<td>Implemented</td>
</tr>
<tr>
<td>Parasoft C/C++test</td>
<td>10.4.2</td>
<td>CERT_C-POS30-a</td>
<td>Avoid overflow due to reading a not zero terminated string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERT_C-POS30-b</td>
<td>The values returned by functions 'read' and 'readlink' shall be used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERT_C-POS30-c</td>
<td>Use of possibly not null-terminated string with functions expecting null-terminated string</td>
</tr>
<tr>
<td>Polyspace Bug Finder</td>
<td>R2019b</td>
<td>CERT C: Rule POS30-C</td>
<td>Checks for misuse of readlink() (rule partially covered)</td>
</tr>
<tr>
<td>PRQA QA-C</td>
<td>9.7</td>
<td>5033</td>
<td></td>
</tr>
</tbody>
</table>

**Related Vulnerabilities**

Search for vulnerabilities resulting from the violation of this rule on the [CERT website](https://www.cert.org).

**Related Guidelines**

**Key here** (explains table format and definitions)

<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Taxonomy item</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE 2.11</td>
<td>CWE-170, Improper null termination</td>
<td>2017-06-13: CERT: Rule subset of CWE</td>
</tr>
</tbody>
</table>

**CERT-CWE Mapping Notes**

**Key here** for mapping notes

**CWE-170 and POS30-C**

CWE-170 = Union( POS30-C, list) where list =

- Non-null terminated strings fed to functions other than POSIX readlink()

**Bibliography**

- Ilja 2006
- Open Group 1997a
- Open Group 2004