strlcpy and strlcat

consistent, safe, string copy and concatenation

Todd C. Miller
<Todd.Miller@cs.colorado.edu>

Theo de Raadt
<deraadt@openbsd.org>

Overview

- Rationale
- What's wrong with using strncpy/strncat?
- How do strlcpy/strlcat help?
- What they don't do
- Implementation
- Who's using them?
- Where to get the code

Rationale

- Buffer overflows have become trivial to exploit
  - Access to source code helps both sides
  - Programmers are eradicating strcpy/strcat from setuid programs
  - Need something to easily replace calls to strcpy/strcat
  - strncpy/strncat are not a good match

Why not use strncpy/strncat?

strncpy/strncat not well suited to size-bounded operations

- Non-intuitive API (lots of people get it wrong)
- Inconsistent use of the length/size parameter
- Difficult to detect truncation
- NUL fill in strncpy( ) has a hidden cost

Why Not (continued)

- strncpy/strncat API non-intuitive
  - Found lots of misuse when auditing OpenBSD
  - Many programmers assume strncpy( ) guarantees NUL-termination--it does not
  - The programmer must clear the last byte manually in case strlen(src) >= sizeof(dst)

Why Not (continued)

- Length parameter used inconsistently
  - For strncpy( ) it is sizeof(dest)
    For strncat( ) it is sizeof(dest) - 1
  - Length parameter for strncat( ) must usually be computed--often incorrectly Eg:
    strncat(path, file, sizeof(path) - strlen(path) - 1);
- Difficult to detect truncation
  - For strncpy, must check strlen(src)
  - For strncat, must save the old length of dst

NUL fill in strncpy( ) has a hidden cost

- Found strncpy( ) of a small string into a 1K buffer to be 3-5 times slower than strcpy( ) depending on the CPU.
  - This is the worst case scenario since you are clearing many more bytes than you copy--but it is also a very common case. Consider copying a pathname info a buffer of size MAXPATHLEN.
  - Probably not just the cost of clearing bytes, but effectively fushing the data cache.
- strlcpy( ) performs almost as well as strcpy( )

How do strlcpy/strlcat help?

size_t strlcat (char *dst, const char *src, size_t siz)
size_t strlcpy (char *dst, const char *src, size_t siz)

- **Consistent, unambiguous interface**
  - **Always NUL-terminate the destination**
  - **Size parameter is the full size of the destination (Eg: `sizeof(buf)`)**
  - **Neither function zero-fills the destination (except for the final NUL to terminate the string).**

How do strlcpy/strlcat help? (continued)

- Both functions provide a useful return value
  - Return the length of the dst string as if there was infinite space
  - For strlcpy( ) this is just strlen(src)
  - For strlcat( ) this is strlen(src) + strlen(orig_dst)
  - Similar to BSD and C9X snprintf( ) return value
  - Makes checking for truncation easy
    If rval >= siz, truncation occurred

What strlcpy/strlcat are not...

- **They are not an attempt to somehow "fix" string handling in C**
  - **If that's what you want there are other options (including C++)**
- **They only operate on normal C strings**
  - **Source string must end in a NUL since we traverse the entire string**
  - **Not usable for strings in** `struct utmp` **for example**

Simplest implementation of strlcpy()

```
size_t strlcpy(char *dst, const char *src,
    size_t siz)
{
   size_t n;
   size_t slen = strlen(src);

   if (siz) {
      if ((n = MIN(slen, siz - 1)))
         memcpy(dst, src, n);
      dst[n] = '\0';
   }
   return(slen);
}
```

Simplest implementation of strlcat()

```
size_t strlcat(char *dst, const char *src,
    size_t siz)
{
   size_t dlen = strlen(dst);

   /* Make sure siz is sane */
   if (dlen < siz - 1)
      return(dlen + strlcpy(dst + dlen,
           src, siz - dlen));
   else
      return(dlen + strlen(src));
}
```

Who's using strlcpy/strlcat?

- **Operating Systems**
  - **First shipped with OpenBSD 2.4**
  - **Approved for inclusion in a future release of Solaris**
- **Applications**
  - **Used by the *rsync* package**
  - **Simple implementation makes it easy to check for the function in a configure script and provide it if needed**

Where to get the code

- OpenBSD 2.5 CD's
- Any OpenBSD ftp mirror
  - pub/OpenBSD/lib/libc/string/strlcpy.c
  - pub/OpenBSD/lib/libc/string/strlcat.c
  - pub/OpenBSD/lib/libc/string/strlcpy.3
- <Todd.Miller@cs.colorado.edu>
- <deraadt@openbsd.org>