User-controlled data may not be null terminated

Name: User-controlled data may not be null terminated

Description: String operations on user-controlled strings can result in buffer overflow or buffer over-read.

ID: cpp/user-controlled-null-termination-tainted

Kind: problem

Severity: warning

Built-in C string functions such as `strcpy` require that their input string arguments are null terminated. If the input string arguments are not null terminated, these functions will read/write beyond the length of the buffer containing the string, resulting in either buffer over-read or buffer overflow, respectively.

Recommendation

Always guard against user-controlled strings that may not be null terminated. Otherwise, an attacker may be able to supply input without null termination.

Example

In this example, a string is read from a file using the `read` function. Because the value may be user-controlled, the string may not be null terminated, in which case the call to `strcpy` will perform a buffer overwrite on `cpy`, potentially resulting in a program crash.

```
1 char buf[BUF_SIZE];
2 read(fd, buf, BUF_SIZE);
3 char cpy[BUF_SIZE];
4 strcpy(cpy, buf);
```

In the revised example, the number of bytes read is recorded in the variable `count`. In addition to checking that the read succeeds (the condition `count >= 0`), a null terminator is inserted before the call to `strcpy`.

```
1 char buf[BUF_SIZE];
2 int count = read(fd, buf, BUF_SIZE);
3 if (count >= 0) {
4   buf[count < BUF_SIZE ? count : BUF_SIZE - 1] = '\0';
5   char cpy[BUF_SIZE];
6   strcpy(cpy, buf);
7 }
```

References

- B. Chess and J. West, *Secure Programming with Static Analysis*, 6.2 Maintaining the Null Terminator. Addison-Wesley. 2007.
- Linux Programmer's Manual: READ(2), STRCPY(3).
- Common Weakness Enumeration: CWE-170.