Software Assurance Tips

A product of the Software Assurance Tips Team[1]

Jon Hood

Monday 26th July, 2021

1 Compounding a Classic TOCTOU Mistake

Updated Friday 6th August, 2021

One project was marked for an operational buffer overflow (CWE-119) and poor coding practices when a buffer reading in a configuration file was set to a static size before reading in the file. The code sent for validation to fix the issue looked similar to that of Listing 1.

```
1  fseek(fp, 0, SEEK_END);
2  long size = ftell(fp);
3  char *buf = malloc(sizeof(char)*size);
4  fseek(fp, 0, SEEK_SET);
5  int c;
6  for (int i = 0; (c = getc(fp)) != EOF; i++)
7  {
8          if (c == '%')
9                  break;
10         buf[i] = c;
11 }
```

Listing 1: Overflows and TOCTOUs

The developer's intent is to read a file into a buffer up to an expected truncation character, but the attempted fixes to the original issues caused more problems than the original code.

First, the code is subject to a Time-Of-Check Time-Of-Use (TOCTOU) issue. By setting up a file system watcher event, an attacker could append a pernicious payload to the end of the file so that the file is larger than `ftell()` originally reported. This causes a buffer overflow (CWE-120) on Line 10.

Second, the developer created a possible wrap-around issue (CWE-190) when the file's character count is larger than `INT_MAX`. The software would then operate outside of the intended buffer boundary on Line 10 (CWE-119).

Finally, while inspecting the operational environment, it was discovered that the configuration file had world write privileges (CWE-276) which could be used to exploit the TOCTOU issue. Coincidentally, the file also had cleartext passwords (CWE-256) for a connected device.

What a mess a single TOCTOU issue can uncover!

References

[1] Jon Hood, ed. SwATips. https://www.SwATips.com/.