New issue

Jump to bottom

Memory-leak bug in printfileinfo, in printinfo.c #60

Open Zzero00 opened this issue on Feb 2 · 1 comment

Zzero00 commented on Feb 2 • edited

There exists one Memory-leak bug in printfileinfo, in printinfo.c, which allows an attacker to leak the address of heap or libc via a crafted file.
To reproduce with the attached poc file:
poc.zip

Heap address leak:
./sfinfo ./heapleak_poc.aiff

Result(See the output of Copyright):

```
$ ./sfinfo ./heapleak_poc.aiff
File Name      ./heapleak_poc.aiff
File Format    Audio Interchange File Format (aiff)
Data Format    unknown
Audio Data     0 bytes begins at offset 0 (0 hex)
               0 channel, -1 frames
Sampling Rate  0.00 Hz
Duration       -inf seconds
Copyright      C��U
```

Libc address leak:
./sfinfo ./libleak_poc.aiff

Result(See the output of Copyright):

```
$ ./sfinfo ./libleak_poc.aiff
File Name      ./libleak_poc.aiff
File Format    Audio Interchange File Format (aiff)
Data Format    unknown
Audio Data     0 bytes begins at offset 0 (0 hex)
               0 channel, -1 frames
Sampling Rate  0.00 Hz
```

```
Duration       -inf seconds
Copyright      Copyright 1991, (d��i
```

This vulnerability can be triggered anywhere the printfileinfo function is called, for example, sfconvert.

The poc.py will help you to calculate the address, which is test on Ubuntu 20.04, python2.

Usage of poc.py:

```
$ python2 poc.py heap
[+] Starting local process './sfinfo': pid 17868
[*] Process './sfinfo' stopped with exit code 0 (pid 17868)
[+] heap_leak:0x55b2425d4243
[+] heap_base:0x55b2425c2000
$ python2 poc.py lib
[+] Starting local process './sfinfo': pid 17920
[*] Process './sfinfo' stopped with exit code 0 (pid 17920)
[+] lib_leak:0x7f3d0cbf5428
[+] libaudiofile_base:0x7f3d0cbc9000
[+] libc_base:0x7f3d0c9bf000
```

The audiofile project is built with:

```
$ ./autogen.sh --disable-docs --prefix=OUTPUT_DIR
$ make
$ make install
```

Descrtption of the Vulnerability:

First, the printfileinfo function calls the copyrightstring function to get data:

```
//printfileinfo function, printinfo.c
bool printfileinfo (const char *filename){
...
char *copyright = copyrightstring(file);
        if (copyright)
        {
                printf("Copyright      %s\n", copyright);
                free(copyright);
        }
...
}
```

Second, the copyrightstring function obtains copyright information from the file and returns a string pointer:

```
//copyrightstring function, printinfo.c
static char *copyrightstring (AFfilehandle file){
```

```
...
int datasize = afGetMiscSize(file, miscids[i]);
                char *data = (char *) malloc(datasize);
                afReadMisc(file, miscids[i], data, datasize);
                copyright = data;
                break;
...
}
```

However, it forgets to use memset or zero bytes to prevent the Memory-Leak Vulnerability.
Most importantly, the attacker can control the length of the memcpy when copying the copyright string, in the afReadMisc function, in Miscellaneous.cpp:

```
//afWriteMisc function, Miscellaneous.cpp
int afWriteMisc (AFfilehandle file, int miscellaneousid, const void *buf, int bytes)
{
...
        int localsize = std::min(bytes,
                miscellaneous->size - miscellaneous->position);
        memcpy((char *) miscellaneous->buffer + miscellaneous->position,
                buf, localsize);
        miscellaneous->position += localsize;
        return localsize;
...
}
```

carnil commented on Feb 22

It appears that a CVE has been assigned to this issue: CVE-2022-24599

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

---

2 participants