Talos Vulnerability Report

TALOS-2022-1533

# ESTsoft Alyac OLE header Mini FAT sectors integer overflow

AUGUST 3, 2022

## CVE NUMBER

CVE-2022-29886

## SUMMARY

An integer overflow vulnerability exists in the way ESTsoft Alyac 2.5.8.544 parses OLE files. A specially-crafted OLE file can lead to a heap buffer overflow, which can result in arbitrary code execution. An attacker can provide a malicious file to trigger this vulnerability.

## CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

ESTsoft Alyac 2.5.8.544

## PRODUCT URLS

Alyac - https://www.estsecurity.com/public/product/alyac

## CVSSV3 SCORE

7.3 - CVSS:3.0/AV:L/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H

## CWE

CWE-680 - Integer Overflow to Buffer Overflow

## DETAILS

Alyac is an antivirus program for Microsoft Windows, developed by ESTsecurity, which is part of ESTsoft.

When Alyac is scanning an OLE formatted file with signature `D0 CF 11 E0 A1 B1 1A E1`, it executes function `sub_180080040` to parse `Number of Mini FAT sectors` field in the header.

```
.text:000000018008005B                 mov     [rsp+1048h+var_28], rax
.text:0000000180080063                 mov     rax, [rcx+OLEParser.file_contents_] ;
[1]
.text:0000000180080067                 mov     rbx, rcx
.text:000000018008006A                 xor     r15d, r15d
.text:000000018008006D                 mov     ecx, [rax+40h]                      ;
[2] file base + 40h -> Number of Mini FAT sectors
.text:0000000180080070                 imul    ecx, [rbx+OLEParser.size_] ; Size   ;
[3] sector size * number of sectors
.text:0000000180080074                 cmp     rcx, [rbx+OLEParser.file_size_]
.text:0000000180080078                 jbe     short loc_180080083
.text:000000018008007A                 lea     eax, [r15+9]
.text:000000018008007E                 jmp     loc_1800801B6
```

In the beginning of the function, the memory address that stores the contents of the file is copied to `RAX` register [1] and is used to get the value of `Number of Mini FAT sectors`, which is at the offset `+40h` from the base of the file [2].

Number of sectors is multiplied by sector size to calculate the size of a heap memory to store contents of sectors [3]. Here, multiplication of two 32-bit unsigned integers is stored to `ECX` register, causing integer overflow. This overflowed value is used as size when allocating a new heap memory [5].

```
.text:0000000180080083                 mov     [rsp+1048h+arg_8], rbp
.text:000000018008008B                 mov     [rsp+1048h+arg_10], rsi
.text:0000000180080093                 mov     [rsp+1048h+arg_18], rdi
.text:000000018008009B                 mov     [rsp+1048h+var_18], r14
.text:00000001800800A3                 call    j_??2@YAPEAX_K@Z ; operator
new(unsigned __int64)    ; [5]
```

Later in the function, there is a loop that copies each sector to newly allocated heap memory [6].

```
.text:0000000180080130                lea     rdx, [rsp+1048h+Src] ; Src
.text:0000000180080135                mov     r8d, r14d       ; Size
.text:0000000180080138                add     rdi,
[rbx+OLEParser.minifat_sectors_buf]
.text:000000018008013C                mov     rcx, rdi        ; void *
.text:000000018008013F                call    memcpy          ; [6]
```

However, the allocated buffer is not large enough to store `sector size * number of sectors` because the size value has been overflowed to a small number when allocating the memory. Therefore repeated copies of each sector will eventually overflow the allocated heap memory.

Following is the crash stack trace when executing the maliciously-crafted OLE file.

```
# Child-SP          RetAddr               Call Site
00 00000031`9eafe048 00007ff9`f17e0144     VCRUNTIME140!memcpy+0x1e3
01 00000031`9eafe050 00007ff9`f17df966     coen!Coen_Clean+0x6c1b4
02 00000031`9eaff0a0 00007ff9`f17d57fa     coen!Coen_Clean+0x6b9d6
03 00000031`9eaff0e0 00007ff9`f17791c2     coen!Coen_Clean+0x6186a
04 00000031`9eaff420 00007ff9`f1765795     coen!Coen_Clean+0x5232
05 00000031`9eaff5a0 00007ff9`f1773974     coen+0x5795
06 00000031`9eaff6e0 00007ff7`ed3f116b     coen!Coen_ScanPath+0xb4
```

## TIMELINE

2022-06-15 - Vendor Disclosure

2022-08-03 - Public Release

2022-08-03 - Vendor Patch Release

## CREDIT

Discovered by Jaewon Min of Cisco Talos.