Talos Vulnerability Report

TALOS-2022-1452

# ESTsoft Alyac PE section headers out of bounds read

MAY 10, 2022

## CVE NUMBER

CVE-2022-21147

## SUMMARY

An out of bounds read vulnerability exists in the malware scan functionality of ESTsoft Alyac 2.5.7.7. A specially-crafted PE file can trigger this vulnerability to cause denial of service and termination of malware scan. 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 2.5.7.7

## PRODUCT URLS

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

## CVSSV3 SCORE

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

## CWE

CWE-823 - Use of Out-of-range Pointer Offset

## DETAILS

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

There exists a vulnerability in a module called `coen.aym` used by Alyac when scanning PE executable files. Module `coen.aym` is reponsible for loading different engines and modules for handling archive formats and initiating the scan. the

While parsing the malformed PE executable file, function `sub_180086C30` is called. This function tries to locate the section header of `.text` section by parsing the PE file. At [1] below, `rax` register stores the address of the `PE\0\0` signature.

```
.text:0000000180086C69 loc_180086C69:                          ; CODE XREF:
sub_180086C30+24↑j
.text:0000000180086C69                                         ; sub_180086C30+2A↑j
.text:0000000180086C69                 mov     rax, [rdi+10h]
[1]
.text:0000000180086C6D
.text:0000000180086C6D loc_180086C6D:                          ; DATA XREF:
.rdata:0000000180405CAC↓o
.text:0000000180086C6D                                         ;
.rdata:0000000180405CC8↓o ...
.text:0000000180086C6D                 mov     [rsp+28h+arg_0], rbx
.text:0000000180086C72                 xor     ebx, ebx
.text:0000000180086C74                 mov     [rsp+28h+arg_8], rsi
.text:0000000180086C79                 mov     [rsp+28h+arg_10], r14
.text:0000000180086C7E                 cmp     bx, [rax+6]     ; Number of sections
[2]
.text:0000000180086C82                 jnb     short loc_180086CC1
.text:0000000180086C84                 mov     r14d, ecx
.text:0000000180086C87                 nop     word ptr [rax+rax+00000000h]
.text:0000000180086C90
.text:0000000180086C90 loc_180086C90:                          ; CODE XREF:
sub_180086C30+8F↓j
.text:0000000180086C90                 lea     rcx, [rbx+rbx*4]
.text:0000000180086C94                 mov     r8, r14         ; MaxCount
.text:0000000180086C97                 lea     rsi, ds:0[rcx*8]
.text:0000000180086C9F                 mov     rdx, rbp        ; String2 ".text"
.text:0000000180086CA2                 mov     rcx, [rdi+18h]  ;
[3]
.text:0000000180086CA6                 add     rcx, rsi        ; String1
SectionHeader->Name
.text:0000000180086CA9                 call    cs:_strnicmp    ; crash!
[4]
.text:0000000180086CAF                 test    eax, eax
.text:0000000180086CB1                 jz      short loc_180086CDD
.text:0000000180086CB3                 mov     rax, [rdi+10h]
.text:0000000180086CB7                 inc     ebx
.text:0000000180086CB9                 movzx   ecx, word ptr [rax+6]
.text:0000000180086CBD                 cmp     ebx, ecx
[5]
.text:0000000180086CBF                 jb      short loc_180086C90
```

Next, it goes into a loop enumerating each section header to find one with the `Name` field set as `.text`. Section table, which follows the PE header, is a list of section headers. Each section header has an 8-byte field at offset +0 to store the name.

`RBX` register is used as the loop counter, which is initialized to 0. It is compared to the `NumberOfSections` field in the file header (part of PE header) at [2,5] so it does not search beyond number of sections. Inside the loop, the offset to the section header is increased by `40` bytes each iteration, which is equal to the size of the section header. The offset is added to `RCX` register, which stores the location of the section table from [3].

While a check is made to make sure only the specified number of section headers is processed, it doesn't check if the PE executable file is big enough to store the number of section headers as defined in the file header. So if given a large number of section headers without `.text` section header, the loop will continue reading until it goes out of bounds of file size, which will eventually cause access violation at [4]. This leads to a crash of the Alyac scanning process, which effectively neutralizes the antivirus scan.

### Crash Information

```
0:018> k
# Child-SP          RetAddr               Call Site
00 00000053`d1c9e3c8 00007ff8`8b286caf     ucrtbase!_ascii_strnicmp+0xe
01 00000053`d1c9e3d0 00007ff8`8b23f8ba     coen!Coen_Clean+0x72d1f
02 00000053`d1c9e400 00007ff8`8b27d32c     coen!Coen_Clean+0x2b92a
03 00000053`d1c9e4c0 00007ff8`8b27cd06     coen!Coen_Clean+0x6939c
04 00000053`d1c9e770 00007ff8`8b263534     coen!Coen_Clean+0x68d76
05 00000053`d1c9e910 00007ff8`8b2191c2     coen!Coen_Clean+0x4f5a4
06 00000053`d1c9ebb0 00007ff8`8b205a50     coen!Coen_Clean+0x5232
07 00000053`d1c9ed30 00007ff8`8b213a4a     coen+0x5a50
08 00000053`d1c9ee70 000001eb`7cddd73e     coen!Coen_ScanHandle+0xba
09 00000053`d1c9eee0 000001eb`7cdc6907     ecm!GetModuleConfigValue+0x64ae
0a 00000053`d1c9ef90 000001eb`7cdfa88e     ecm+0x56907
0b 00000053`d1c9f110 000001eb`7cdd69f0     ecm!GetModuleConfigValue+0x235fe
0c 00000053`d1c9f1f0 00007ff8`d5632a51     ecm!ScanFile+0x40
0d 00000053`d1c9f230 00007ff8`d564c219     scn+0x32a51
0e 00000053`d1c9f350 00007ff8`d564b63f     scn!ForceCloseScan+0xebf9
0f 00000053`d1c9f850 00007ff8`d564ab3d     scn!ForceCloseScan+0xe01f
10 00000053`d1c9fa70 00007ff8`eb6c6c0c     scn!ForceCloseScan+0xd51d
11 00000053`d1c9faa0 00007ff8`ec8654e0     ucrtbase!thread_start<unsigned int
(__cdecl*)(void *),1>+0x4c
12 00000053`d1c9fad0 00007ff8`edec485b     KERNEL32!BaseThreadInitThunk+0x10
13 00000053`d1c9fb00 00000000`00000000     ntdll!RtlUserThreadStart+0x2b
```

### VENDOR RESPONSE

The product was updated to 2.5.7.7

### TIMELINE

2022-01-31 - Vendor disclosure

2022-04-12 - Talos reissued copies of report

2022-04-29 - Vendor patched

2022-05-10 - Public Release

### CREDIT

Discovered by Jaewon Min of Cisco Talos.