Talos Vulnerability Report

TALOS-2021-1308

PowerISO DMG File Format Handler memory corruption vulnerability

JUNE 28, 2021

CVE NUMBER

CVE-2021-21871

Summary

A memory corruption vulnerability exists in the DMG File Format Handler functionality of PowerISO 7.9. A specially crafted DMG file can lead to an out-of-bounds write. An attacker can provide a malicious file to trigger this vulnerability. The vendor fixed it in a bug-release of the current version.

Tested Versions

PowerISO 7.9

Product URLs

https://www.poweriso.com/

CVSSv3 Score

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

CWE

CWE-787 - Out-of-bounds Write

Details

PowerISO is a powerful CD/DVD/BD image file processing tool, which allows to open, extract, burn, create, edit, compress, encrypt, split and convert ISO files, and mount ISO files with an internal virtual drive. Recent versions provide support for Apple Disk Image file format (also known as DMG - file extension).

Apple Disk Image is a disk image format commonly used by the macOS operating system. This format can be structured according to one of several proprietary disk image formats, including the Universal Disk Image Format (UDIF) from Mac OS X and the New Disk Image Format (NDIF) from Mac OS 9. This format is still not officially documented by Apple. However there are various 3rd party resources that provide additional details regarding the Apple Disk Image format.

Typically, when inflating a ZLIB block inside a DMG file, the output buffer size should be limited to `SECTOR_SIZE * BuffersNeeded` (from the `BLKXTable`), but whatever the output buffer size limit is, it should be checked during the inflation operation. PowerISO, while inflating ZLIB section, does not correctly check the boundary of the output buffer. This leads to memory corruption when specifically crafted DMG file is parsed:

```
.text:0000000000006E73 loc_6E73:                              ; CODE XREF: sub_6B20+376↓j
.text:0000000000006E73                 mov     al, [rcx+1]
.text:0000000000006E76                 add     rcx, 3
.text:0000000000006E7A                 add     r8, 3
.text:0000000000006E7E                 mov     [r8-2], al      ; memory corruption 1
.text:0000000000006E82                 mov     al, [rcx-1]
.text:0000000000006E85                 add     r11d, 0FFFFFFFDh
.text:0000000000006E89                 cmp     r11d, 2
.text:0000000000006E8D                 mov     [r8-1], al
.text:0000000000006E91                 mov     al, [rcx]
.text:0000000000006E93                 mov     [r8], al        ; memory corruption 2
.text:0000000000006E96                 ja      short loc_6E73
.text:0000000000006E98                 test    r11d, r11d
.text:0000000000006E9B                 jz      short loc_6EC0
```

Memory for decompression function is provided here:

```
0x71E8: entry func init r11=dest=000000002fdd7b20
0x833E: set dest buff r11=000000002fdd7b2d
0x7F0A: before CALL buggy_inflate entry r11=dest=000000002fdd7b2d max_write=000000002fddfb2d r9=0000000000007ff3

0:000> db 0x2fdd7b2d+0x8000
00000000`2fddfb2d  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fddfb3d  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fddfb4d  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fddfb5d  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

Typical window size is 0x8000 bytes. In a malformed DMG file (malformed ZLIB section) it is possible to add a malformed DIST (waiting for distance code) or DISTEX (waiting for distance extra bits) to the ZLIB bytecode. This leads to further problems with decompression.

In a typical ZLIB implementation, after each cycle of inflate function, there is a safety check present [1]:

```
in -= strm->avail_in;
out -= strm->avail_out;
strm->total_in += in;
strm->total_out += out;
state->total += out;
if (state->wrap && out)
    strm->adler = state->check =
        UPDATE(state->check, strm->next_out - out, out);
strm->data_type = state->bits + (state->last ? 64 : 0) +
                  (state->mode == TYPE ? 128 : 0);

if (((in == 0 && out == 0) || flush == Z_FINISH) && ret == Z_OK) [1]
  ret = Z_BUF_ERROR;
```

Typical ZLIB output when inflating our malformed file:

```
INFLATE
have=0 left=1637
LEN have=0 left=1637
in = 0 (strm->avail_in = 0)
out = 0 (strm->avail_out = 1637)
return ERROR Z_BUF_ERROR
```

`Z_BUF_ERROR` is returned when no progress was possible or if there was not enough room in the output buffer.

In PowerISO's implementation the output buffer length is not verified or is not verified correctly, which leads to memory corruption.

```
PowerISO+0x6e7e:
00000000`00406e7e 418840fe        mov     byte ptr [r8-2],al ds:00000000`2fdd9000=??
0:000> db @r8-2
00000000`2fdd9000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9040  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9050  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9060  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000000`2fdd9070  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

```
0:000> !analyze -v
*******************************************************************************
*                                                                             *
*                        Exception Analysis                                   *
*                                                                             *
*******************************************************************************


KEY_VALUES_STRING: 1

    Key  : AV.Fault
    Value: Write

    Key  : Analysis.CPU.Sec
    Value: 1

    Key  : Analysis.DebugAnalysisProvider.CPP
    Value: Create: 8007007e on I
    Key  : Analysis.DebugData
    Value: CreateObject

    Key  : Analysis.DebugModel
    Value: CreateObject

    Key  : Analysis.Elapsed.Sec
    Value: 81

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 108

    Key  : Analysis.System
    Value: CreateObject

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 19890

    Key  : Timeline.Process.Start.DeltaSec
    Value: 1513


NTGLOBALFLAG:  400

PROCESS_BAM_CURRENT_THROTTLED: 0

PROCESS_BAM_PREVIOUS_THROTTLED: 0

APPLICATION_VERIFIER_FLAGS:  0

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 0000000000406e93 (PowerISO+0x0000000000006e93)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000001
   Parameter[1]: 0000000005b57000
Attempt to write to address 0000000005b57000

FAULTING_THREAD:  0000197c

PROCESS_NAME:  PowerISO.exe

WRITE_ADDRESS:  0000000005b57000

ERROR_CODE: (NTSTATUS) 0xc0000005 - Instrukcja w 0x%p odwo a a si  do pami ci pod adresem 0x%p. Pami   nie mo e by  %s.

EXCEPTION_CODE_STR:  c0000005

EXCEPTION_PARAMETER1:  0000000000000001

EXCEPTION_PARAMETER2:  0000000005b57000

STACK_TEXT:
00000000`0014ddc0 00000000`00407f2a : 00000000`00008000 00000000`01010117 00000000`0000003f 00000000`00008000 : PowerISO+0x6e93
00000000`0014de40 00000000`004092fa : 00000000`00007ff3 00000000`00000000 00000000`00008000 00000000`05000401 : PowerISO+0x7f2a
00000000`0014dee0 00000000`0040afa1 : 00000000`00000000 00000000`05a1c730 00000000`00000bc0 00000000`00000000 : PowerISO+0x92fa
00000000`0014dfb0 00000000`004059a2 : 00000000`00000bc0 00000000`00001000 00000000`00010000 00000000`00010000 : PowerISO+0xafa1
00000000`0014e020 00000000`0041486a : 00000000`00000002 00000000`058de6f0 00000000`00000008 00000000`00000000 : PowerISO+0x59a2
00000000`0014e080 00000000`004167d6 : 00000000`059e3ed0 00000000`00416d44 00000000`00000028 00000000`005d115a : PowerISO+0x1486a
00000000`0014e0f0 00000000`00415b5d : 00000000`00000000 00000000`059cbe30 00000000`00000004 00000000`059e3db0 : PowerISO+0x167d6
00000000`0014e180 00000000`00417518 : 00000000`00000000 00000000`059e3db0 00000000`058de6f0 00000000`00000000 : PowerISO+0x15b5d
00000000`0014e1e0 00000000`00405ea2 : 00000000`00000028 00000000`00000000 00000000`058829a0 00000000`00000000 : PowerISO+0x17518
00000000`0014e260 00000000`004063f8 : 00000000`01188be3 00000000`12b97fcb 00000000`c0340a62 00000000`4609824c : PowerISO+0x5ea2
00000000`0014e310 00000000`005af9db : 00000000`0014e550 00000000`0014e550 00000000`00000001 00000000`02b77d4c : PowerISO+0x63f8
00000000`0014e4c0 00000000`004e507d : 00000000`0014e670 00000000`00000000 00000000`00000000 00000000`00000000 : PowerISO+0x1af9db
00000000`0014e5e0 00000000`004e5949 : 00000000`01188b17 00000000`0014e948 00000000`12b97fcb 00000000`c0340a62 : PowerISO+0xe507d
00000000`0014e910 00000000`00531da3 : 00000000`00000004 00000000`05895e8c 00000000`00000000 00000000`030cb650 : PowerISO+0xe5949
00000000`0014e940 00000000`004e09a1 : 00000000`0014ea80 00000000`00000000 00000000`00000004 00007ff9`00000002 : PowerISO+0x131da3
00000000`0014e970 00000000`005fbce5 : 00000000`00000001 00007ff9`2272c7b8 00000000`00000000 00000000`00503eec : PowerISO+0xe09a1
00000000`0014f890 00000000`005f887f : 00000000`02a50150 00000000`00613ddd 00000000`00000005 00000000`00000080 : PowerISO+0x1fbce5
00000000`0014f9c0 00000000`004e0f37 : 00000000`00000000 00000000`0010046a 00000000`030cb650 00000000`00000001 : PowerISO+0x1f887f
00000000`0014fa20 00000000`005fa008 : ffffffff`fffffffe 00000000`00000113 00000000`00000000 00000000`00000113 : PowerISO+0xe0f37
00000000`0014fa50 00000000`005fa1b6 : 00000000`008b3bc0 00000000`00c4d430 00000000`00000001 00007ff9`02000002 : PowerISO+0x1fa008
00000000`0014fb10 00007ff9`2272e858 : 00000000`008b3b60 00000000`00000113 00000000`00000001 00000000`00000000 : PowerISO+0x1fa1b6
00000000`0014fb70 00007ff9`2272e299 : 00000000`0010046a 00000000`005fa168 00000000`0010046a 00000000`00000113 :
USER32!UserCallWinProcCheckWow+0x2f8
00000000`0014fd00 00000000`005f6295 : 00000000`005fa168 00000000`008b3b60 00000000`00000002 00000000`008b3b60 :
USER32!DispatchMessageWorker+0x249
00000000`0014fd80 00000000`005f60d1 : 00000000`008b3b60 00000000`00400000 00000000`00000001 00000000`00000000 : PowerISO+0x1f6295
00000000`0014fdc0 00000000`005fe64f : 00000000`005c948c 00000000`00000000 00000000`00621258 00000000`00621268 : PowerISO+0x1f60d1
00000000`0014fe20 00000000`005ceeab : 00000000`00000041 00000000`00000000 00000000`00000000 00000000`00400000 : PowerISO+0x1fe64f
00000000`0014fe80 00007ff9`22dc7034 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : PowerISO+0x1ceeab
00000000`0014ff30 00007ff9`24062651 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
KERNEL32!BaseThreadInitThunk+0x14
00000000`0014ff60 00000000`00000000 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
ntdll!RtlUserThreadStart+0x21


STACK_COMMAND:  ~0s ; .cxr ; kb

SYMBOL_NAME:  PowerISO+6e93

MODULE_NAME: PowerISO
```

```
IMAGE_NAME:  PowerISO.exe

FAILURE_BUCKET_ID:  INVALID_POINTER_WRITE_c0000005_PowerISO.exe!Unknown

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x64

OSNAME:  Windows 10

FAILURE_ID_HASH:  {1b12d601-7fad-79d8-d5a8-9f7caedc20c8}

Followup:     MachineOwner
---------
```

Timeline

2021-06-05 - Vendor Disclosure
2021-06-28 - Public Release

CREDIT

Discovered by Piotr Bania of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1277

NEXT REPORT TALOS-2021-1283