Death by Thumb Drive

File System Fuzzing with CERT BFF

Will Dormann
Senior Vulnerability Analyst
CERT/CC

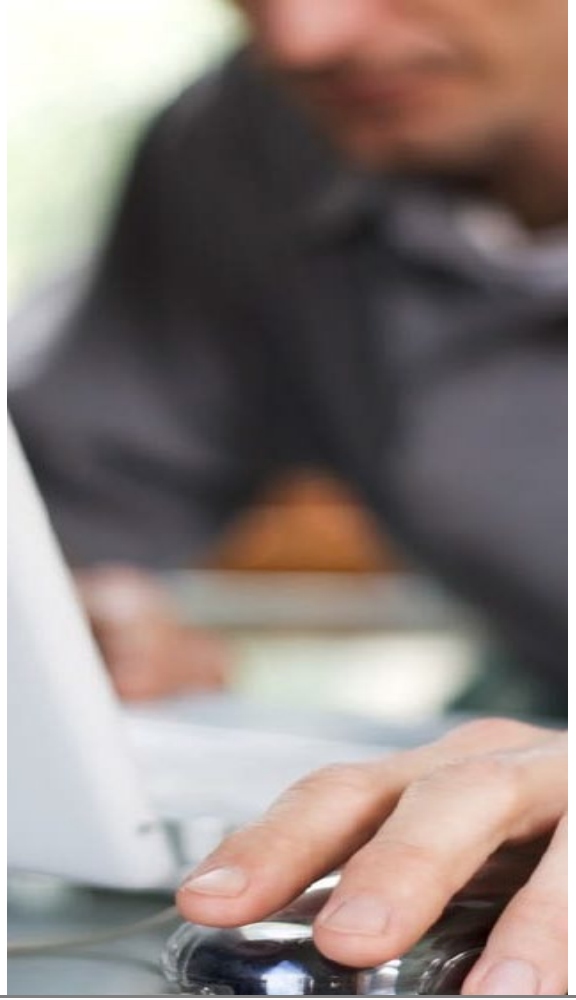
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM19-0594

Death by Thumb Drive

Fuzzing Filesystems with BFF



Extending BFF Fuzzing

For each mutated file, you can run a shell script to do **whatever you want** to the file you just fuzzed.

```
target:
    program: /usr/bin/file
    cmdline_template: $PROGRAM $SEEDFILE
    copyfuzzedto: /home/test/fs.bin
    postprocessfuzzed: /home/test/testdisk.sh
```

What testdisk.sh can do

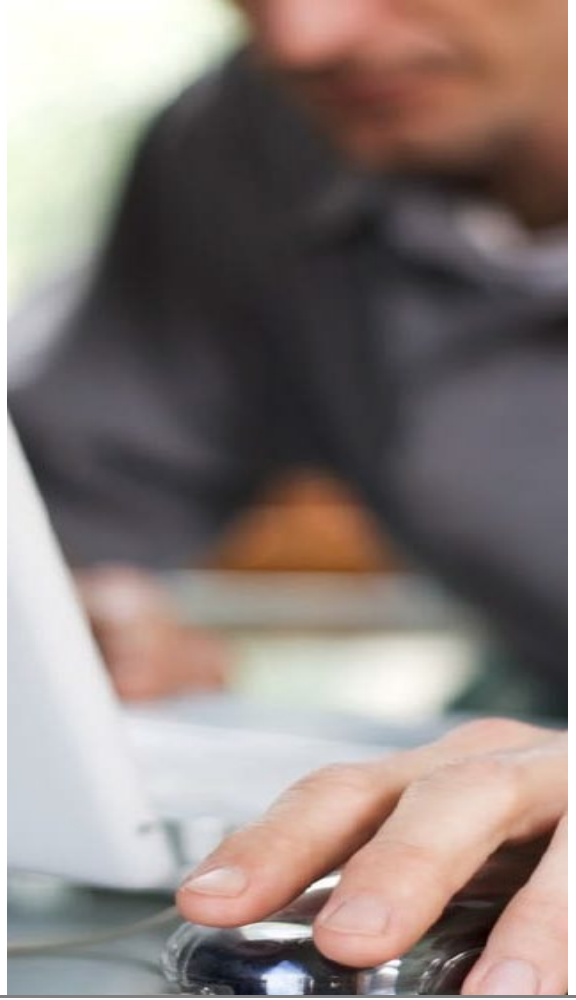
```
rootdisk=$(mount | grep "on / " | awk '{print $1}')
if [ "$rootdisk" = "/dev/sda2" ]; then
  usbdisk=sdb
else
  usbdisk=sda
fi
dd if=/home/test/fs.bin of=/dev/$usbdisk bs=10M
partprobe -s /dev/$usbdisk
mount /dev/${usbdisk}1 /mnt/usb
find /mnt/usb
tar cvf /dev/null /mnt/usb
umount /mnt/usb
```

Eventually

```
[ 1244.616676] BUG: unable to handle kernel NULL pointer dereference at 0000000000000018
[ 1244.624092] PGD 0 P4D 0
[ 1244.625489] Oops: 0000 [#1] SMP NOPTI
[ 1244.632268] CPU: 0 PID: 10637 Comm: mount Kdump: loaded Not tainted 4.20.13 #3
[ 1244.639026] Hardware name: VMware, Inc. VMware7,1/440BX Desktop Reference Platform, BIOS VMW71.00V.0.B64.1508272355 08/27/201
5
[ 1244.648919] RIP: 0010:journal_init+0x109b/0x1670 [reiserfs]
[ 1244.653004] Code: 8b 85 50 ff ff ff 42 8b 74 b0 0c 48 8b bb d0 00 00 00 8b 53 18 b9 08 00 00 00 e8 10 31 2c dd 49 89 45 00 48
 8b 8b d8 03 00 00 <4c> 8b 68 18 48 8b 79 08 8b 07 49 39 c5 0f 87 ce 03 00 00 48 8b 41
[ 1244.666585] RSP: 0018:ffffc90002a3fbb0 EFLAGS: 00010286
[ 1244.669280] RAX: 0000000000000000 RBX: ffff888027cb2000 RCX: ffff888027cc0a00
[ 1244.679365] RDX: 0000000000000000 RSI: ffff88807a01fb80 RDI: ffffea00008129c0
[ 1244.683459] RBP: ffffc90002a3fcb8 R08: 0000000000000000 R09: ffff88807a501c80
[ 1244.690394] R10: 0000000000000000 R11: 000015ffff7ed63f R12: 000000000000000c
[ 1244.697517] R13: ffff888027c74c60 R14: 000000000000000c R15: ffff888027c74460
[ 1244.703969] FS:  00007f18f5a14080(0000) GS:ffff88807aa00000(0000) knlGS:0000000000000000
[ 1244.711510] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[ 1244.717399] CR2: 0000000000000018 CR3: 000000007e0f8000 CR4: 00000000000406f0
[ 1244.724164] Call Trace:
[ 1244.726094]  reiserfs_fill_super+0x4c2/0xca0 [reiserfs]
[ 1244.737688]  ? snprintf+0x45/0x70
[ 1244.740106]  mount_bdev+0x17f/0x1b0
[ 1244.751256]  ? finish_unfinished+0x680/0x680 [reiserfs]
[ 1244.754699]  get_super_block+0x15/0x20 [reiserfs]
[ 1244.756121]  mount_fs+0x37/0x150
[ 1244.766518]  vfs_kern_mount.part.26+0x5d/0x110
[ 1244.769301]  do_mount+0x5ed/0xce0
[ 1244.775523]  ? memdup_user+0x4f/0x80
[ 1244.782106]  ksys_mount+0x98/0xe0
[ 1244.783564]  __x64_sys_mount+0x25/0x30
[ 1244.792971]  do_syscall_64+0x5a/0x120
[ 1244.797763]  entry_SYSCALL_64_after_hwframe+0x44/0xa9
[ 1244.810625] RIP: 0033:0x7f18f52c23ca
[ 1244.812554] Code: 48 8b 0d c1 8a 2c 00 f7 d8 64 89 01 48 83 c8 ff c3 66 2e 0f 1f 84 00 00 00 00 00 0f 1f 44 00 00 49 89 ca b8
 a5 00 00 00 0f 05 <48> 3d 01 f0 ff ff 73 01 c3 48 8b 0d 8e 8a 2c 00 f7 d8 64 89 01 48
[ 1244.826709] RSP: 002b:00007ffec2482008 EFLAGS: 00000202 ORIG_RAX: 00000000000000a5
[ 1244.828020] RAX: ffffffffffffffda RBX: 0000556b2d142a40 RCX: 00007f18f52c23ca
[ 1244.839582] RDX: 0000556b2d14cb80 RSI: 0000556b2d142c40 RDI: 0000556b2d142c20
[ 1244.843272] RBP: 0000000000000000 R08: 0000000000000000 R09: 00007f18f530e1b0
[ 1244.844546] R10: 00000000c0ed0000 R11: 0000000000000202 R12: 0000556b2d142c20
[ 1244.856746] R13: 0000556b2d14cb80 R14: 0000000000000000 R15: 00007f18f57ea8a4
[ 1244.857983] Modules linked in: reiserfs hfs f2fs ntfs nilfs2 minix hfsplus xfs nls_utf8 isofs ufs nfsv3 nfs_acl rpcsec_gss_kr
b5 auth_rpcgss nfsv4 nfs lockd grace fscache nls_iso8859_1 vmw_balloon crct10dif_pclmul crc32_pclmul ghash_clmulni_intel aesni_i
ntel aes_x86_64 crypto_simd cryptd glue_helper serio_raw vmw_vmci sunrpc sch_fq_codel ip_tables x_tables autofs4 btrfs xor zstd_
compress raid6_pq libcrc32c drm_kms_helper syscopyarea sysfillrect sysimgblt fb_sys_fops ttm drm psmouse e1000 i2c_piix4 i2c_cor
e ahci vmw_pvscsi libahci pata_acpi floppy
[ 1244.905050] CR2: 0000000000000018
```

Death by Thumb Drive

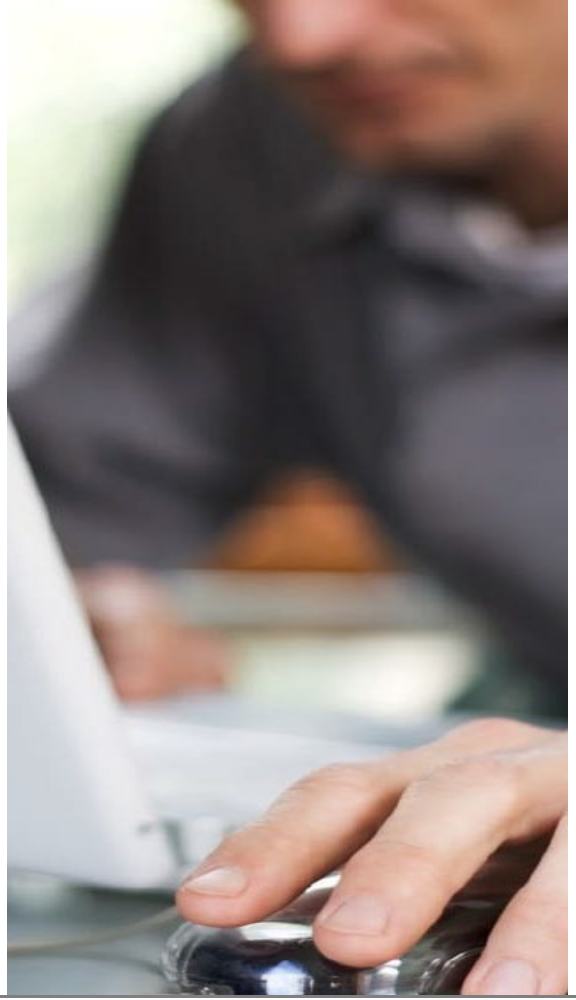
Investigating Crashes



Death by Thumb Drive

Investigating Crashes

Linux



Linux - Background

I happen to believe that not having a kernel debugger forces people to think about their problem on a different level than with a debugger. I think that without a debugger, you don't get into that mindset where you know how it behaves, and then you fix it from there. Without a debugger, you tend to think about problems another way. You want to understand things on a different _level_.

Because I'm a bastard, and proud of it!

Linus Torvalds - Wed, 6 Sep 2000

Read more about Linux's background here.

Linux Kernel Debugging

Linux kernel crash debugging can be done via gdb over a serial port.

- Slow
- Unreliable

Remote gdb Over Serial

```
$ sudo gdb /usr/lib/debug/boot/vmlinux* -baud 115200

>>> target remote /dev/ttyS1

── Assembly ──────────────────────────────────────────────────────────────────
Cannot access memory at address 0x0

── Registers ─────────────────────────────────────────────────────────────────
   rax 0x0000000000000000          rbx 0xffff9c4043063500          rcx 0x0000000074746178
   rdx 0x0000000000000000          rsi 0xffff9c4043063500          rdi 0xffff9c4043305670
   rbp 0xffffb47005207c50          rsp 0xffffb47005207c20           r8 0x0000000073727474
    r9 0x0000000000000000          r10 0xffffb47005207ae0          r11 0x0000000000000000
   r12 0xffff9c4043063140          r13 0xffffb47005207c60          r14 0x0000000000000000
   r15 0xffff9c40b48c4400          rip 0x0000000000000000       eflags [ PF ZF IF RF ]
    cs 0x00000010                   ss 0x00000018                   ds 0x00000000
    es 0x00000000                   fs 0x00000000                   gs 0x0000000b

>>> bt

#0  0x0000000000000000 in irq_stack_union ()
#1  0xffffffff8588441a in ?? ()
#2  0xffff9c4043063140 in ?? ()
#3  0xffffffffc06f23a8 in ?? ()
#4  0xffffffffc06f23a8 in ?? ()
```

Automated Coredumps with linux-crashdump

gdb over serial is too much manual work. We can do better: Linux-crashdump can be read here.

Linux-crashdump transitions to a separate kernel for debugging if the running kernel crashes.

Problem: Linux-crashdump doesn't work by default on Ubuntu 18.04

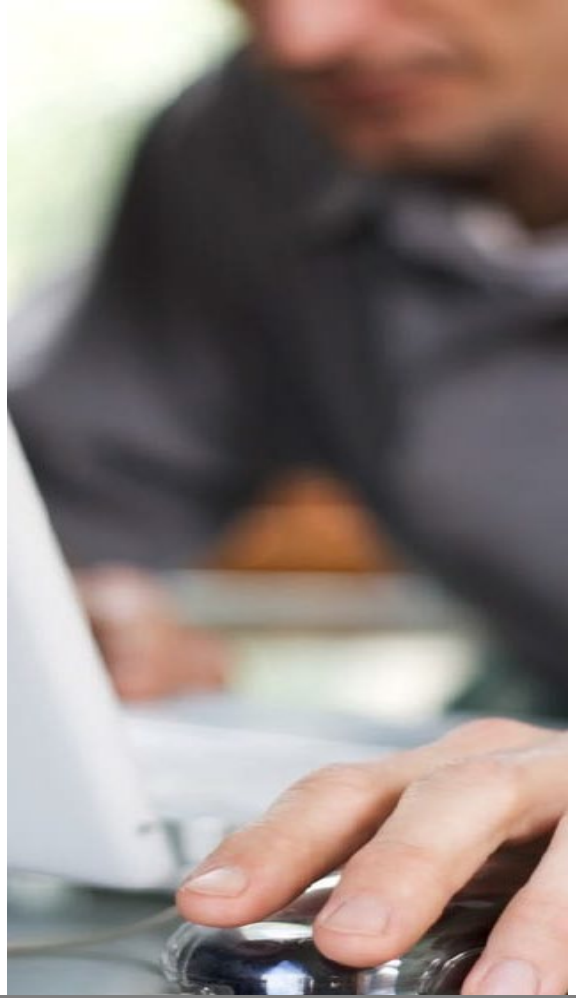
Fix: Modify `/etc/default/grub.d`

```
GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT crashkernel=384M-:256M"
```

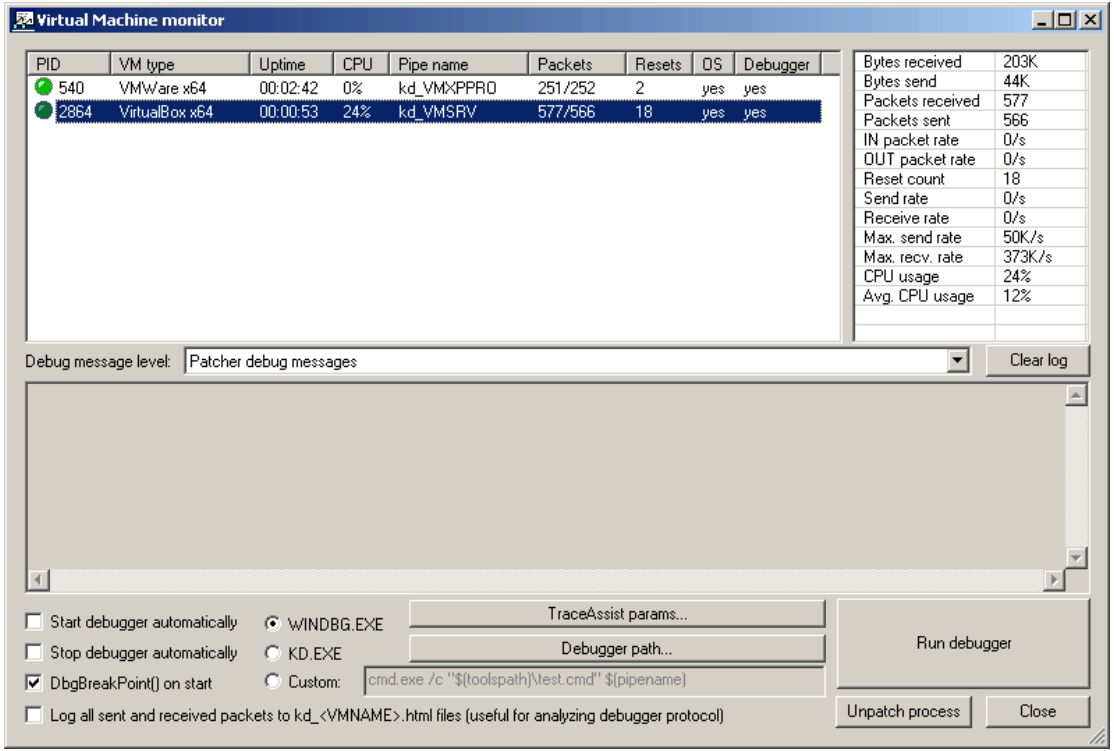
Death By Thumb Drive

Investigating Crashes

Windows

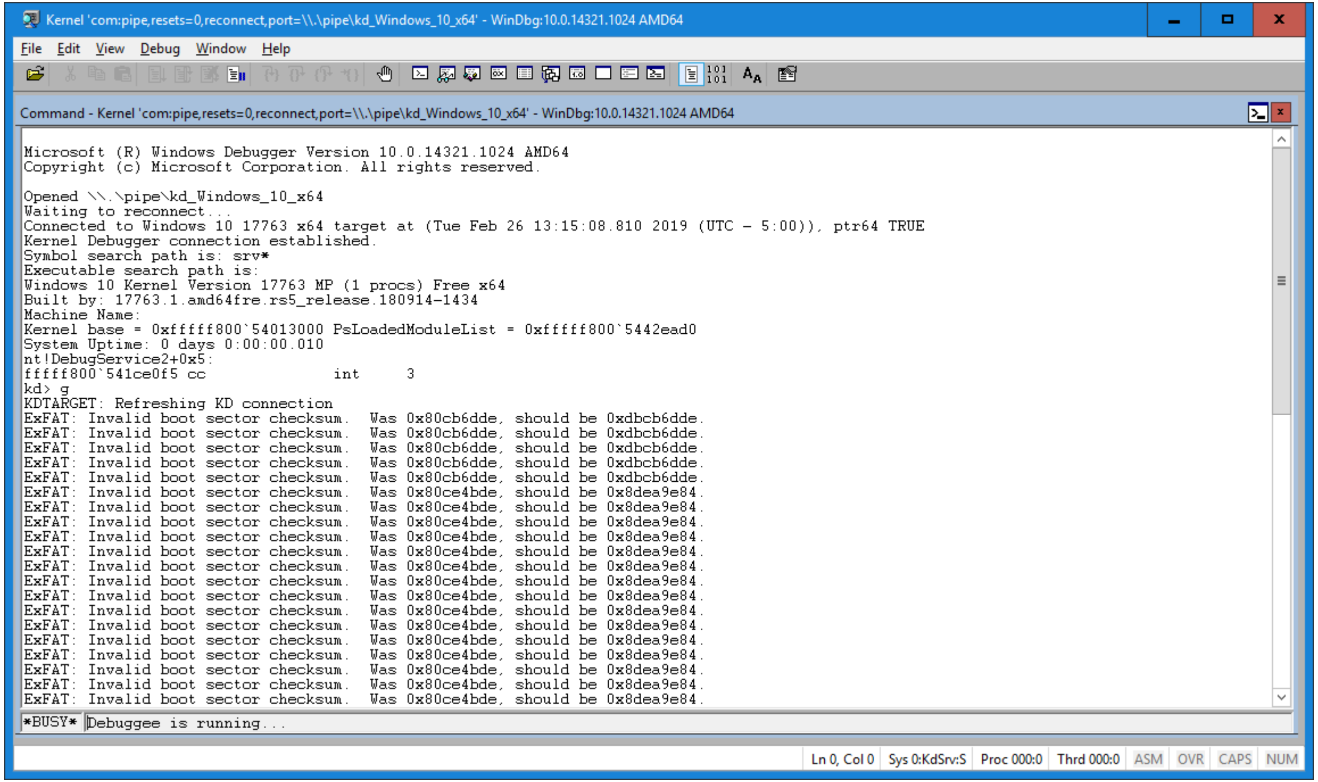


VirtualKD



[Read more about the VirtualKD here.]

VirtualKD in Action



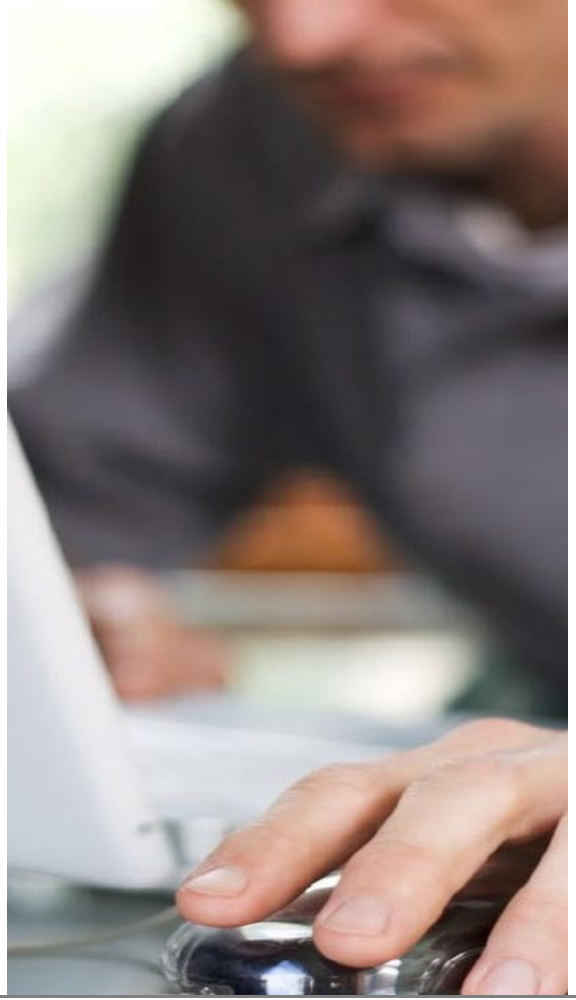
Eventually…

```
1: kd> .load msec
1: kd> !exploitable -v

<SNIP>

Description: Write Access Violation in Kernel Memory
Short Description: WriteAV
Exploitability Classification: EXPLOITABLE
Recommended Bug Title: Exploitable - Write Access Violation in Kernel
Memory starting at nt!DbgBreakPointWithStatus+0x0000000000000000
(Hash=0xa192536f.0xb8bb4599)
```

Death By Thumb Drive

Investigating Crashes

macOS



Configuring macOS for Kernel Debugging

Read how to set up the kernel debugger here.

1. Install the macOS Kernel Debug Kit
2. Update nvram

```
$ sudo nvram boot-args="debug=0x141 kext-dev-mode=1
kcsuffix=development pmuflags=1 -v"
```

When a Crash is Encountered

```
$ lldb
/Library/Developer/KDKs/KDK_10.13.6_17G6009.kdk/System
/Library/Kernels/kernel.development

(lldb) kdp-remote 192.168.0.188

Version: Darwin Kernel Version 17.7.0: Sun Jan 27
13:29:50 PST 2019; root:xnu-
4570.71.27~1/DEVELOPMENT_X86_64; UUID=062F2465-64E9-
332A-9E37-F76C50D9C2CE; stext=0xffffff8001200000

(lldb) bt
```

When a Crash is Encountered

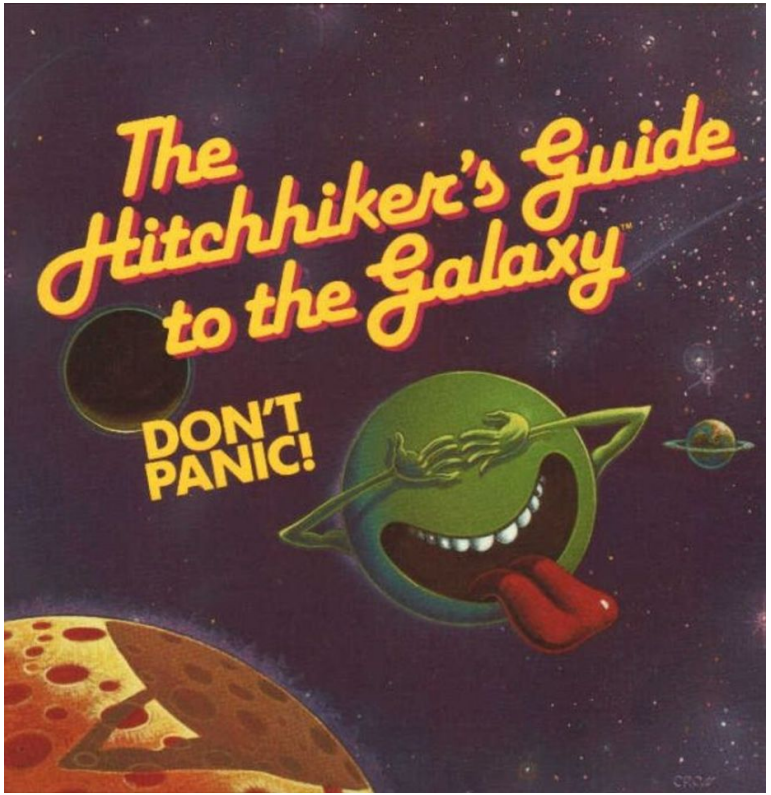
```
* thread #1, stop reason = signal SIGSTOP
  * frame #0: 0xffffff800137ba7a kernel.development`panic_trap_to_debugger [inlined]
current_cpu_datap at cpu_data.h:401 [opt]
    frame #1: 0xffffff800137ba7a kernel.development`panic_trap_to_debugger [inlined]
current_processor at cpu.c:220 [opt]
    frame #2: 0xffffff800137ba7a kernel.development`panic_trap_to_debugger [inlined]
DebuggerTrapWithState(db_op=DBOP_PANIC, db_message=<unavailable>, db_panic_str="\"%s():
data1_len <
sizeof(FILENAME_ATTR)\\n\"@/BuildRoot/Library/Caches/com.apple.xbs/Sources/ntfs/ntfs-
94/kext/ntfs_collate.c:102", db_panic_args=0xffffff8061103760, db_panic_options=0,
db_proceed_on_sync_failure=1, db_panic_caller=18446743521867916843) at debug.c:463 [opt]
    frame #3: 0xffffff800137ba4a
kernel.development panic_trap_to_debugger(panic_format_str="\"%s(): data1_len <
sizeof(FILENAME_ATTR)\\n\"@/BuildRoot/Library/Caches/com.apple.xbs/Sources/ntfs/ntfs-
94/kext/ntfs_collate.c:102", panic_args=0xffffff8061103760, reason=0, ctx=0x0000000000000000,
panic_options_mask=0, panic_caller=18446743521867916843) at debug.c:724 [opt]
    frame #4: 0xffffff800137b84c kernel.development`panic(str=<unavailable>) at debug.c:611
[opt]
    frame #5: 0xffffff7f83ace62b
```

Wait, a Panic?

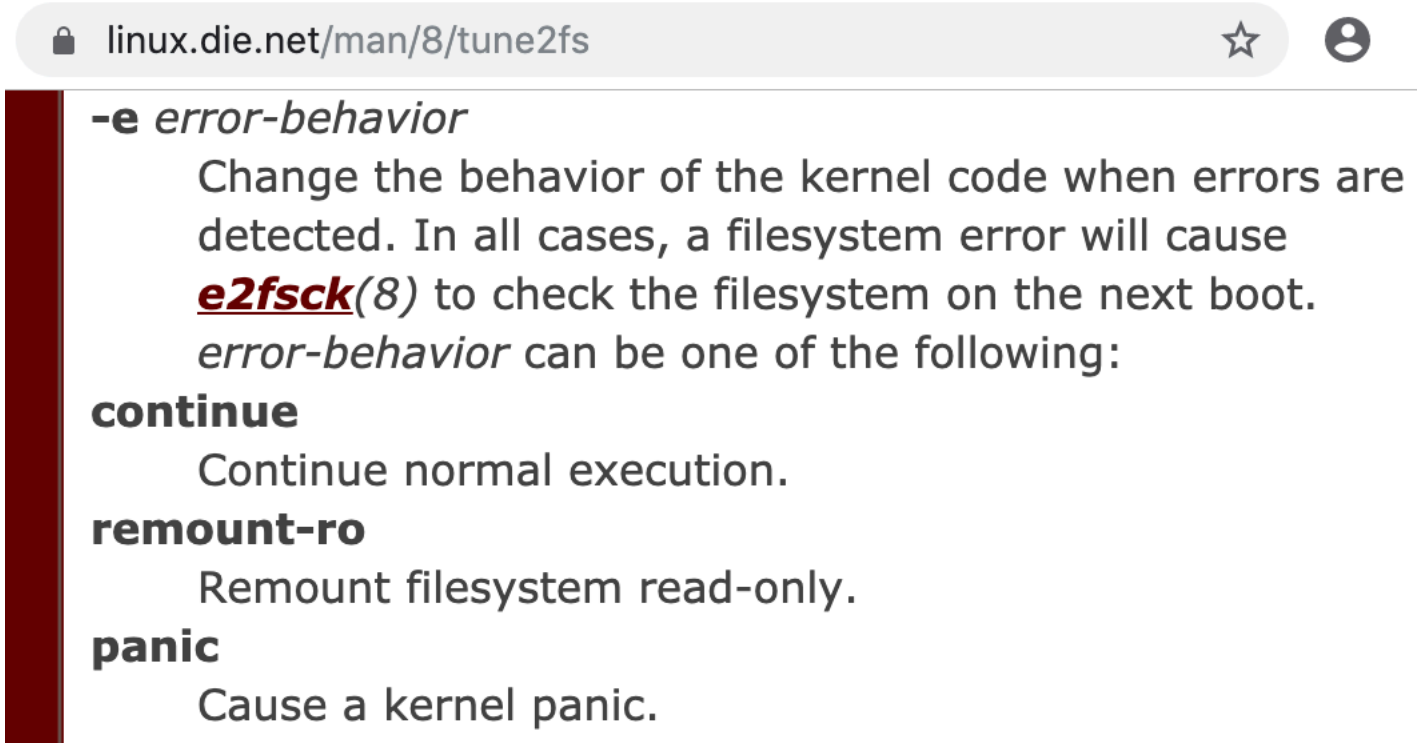
Find the open-source code here.

```
/**
 * ntfs_collate_filename - filename collation
 *
 * Used for COLLATION_FILENAME.
 *
 * Note: This only performs exact matching as it is only intended to be used
 * when looking up a particular name that is already known to exist and we just
 * want to locate the correct index entry for it so that we can modify/delete
 * it.  Alternatively, we want to add a new name and we already know that it
 * does not exist in the index so we just want to locate the correct index
 * entry in front of which we need to insert the name.
 */
static int ntfs_collate_filename(ntfs_volume *vol,
                const void *data1, const int data1_len,
                const void *data2, const int data2_len)
{
        const FILENAME_ATTR *fn1 = data1;
        const FILENAME_ATTR *fn2 = data2;
        int rc;

        ntfs_debug("Entering.");
        if (data1_len < (int)sizeof(FILENAME_ATTR))
                panic("%s(): data1_len < sizeof(FILENAME_ATTR)\n",
                                __FUNCTION__);
        if (data2_len < (int)sizeof(FILENAME_ATTR))
                panic("%s(): data2_len < sizeof(FILENAME_ATTR)\n",
                                __FUNCTION__);
        /*
```

Perhaps Don’t Panic?

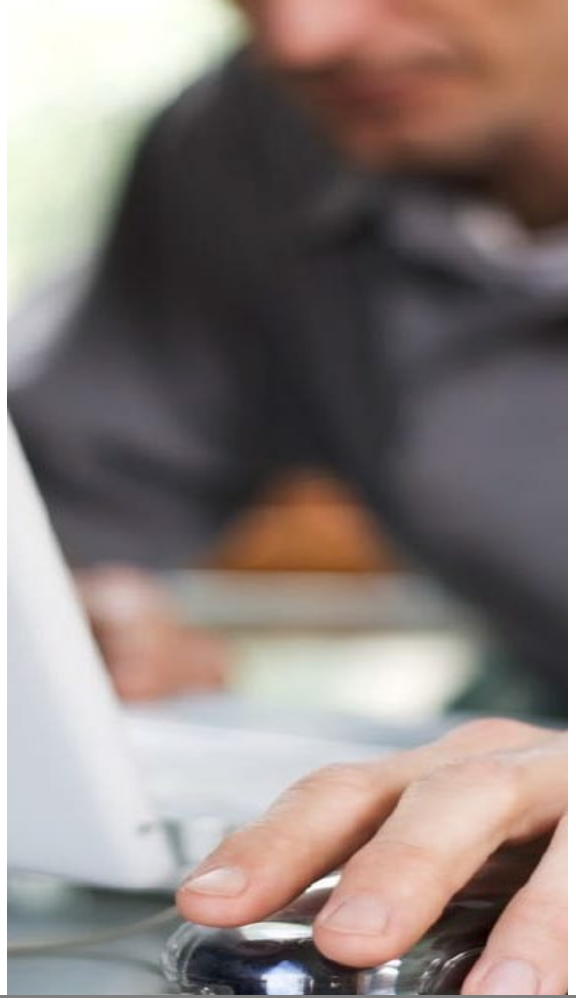


Linux Panics by Design



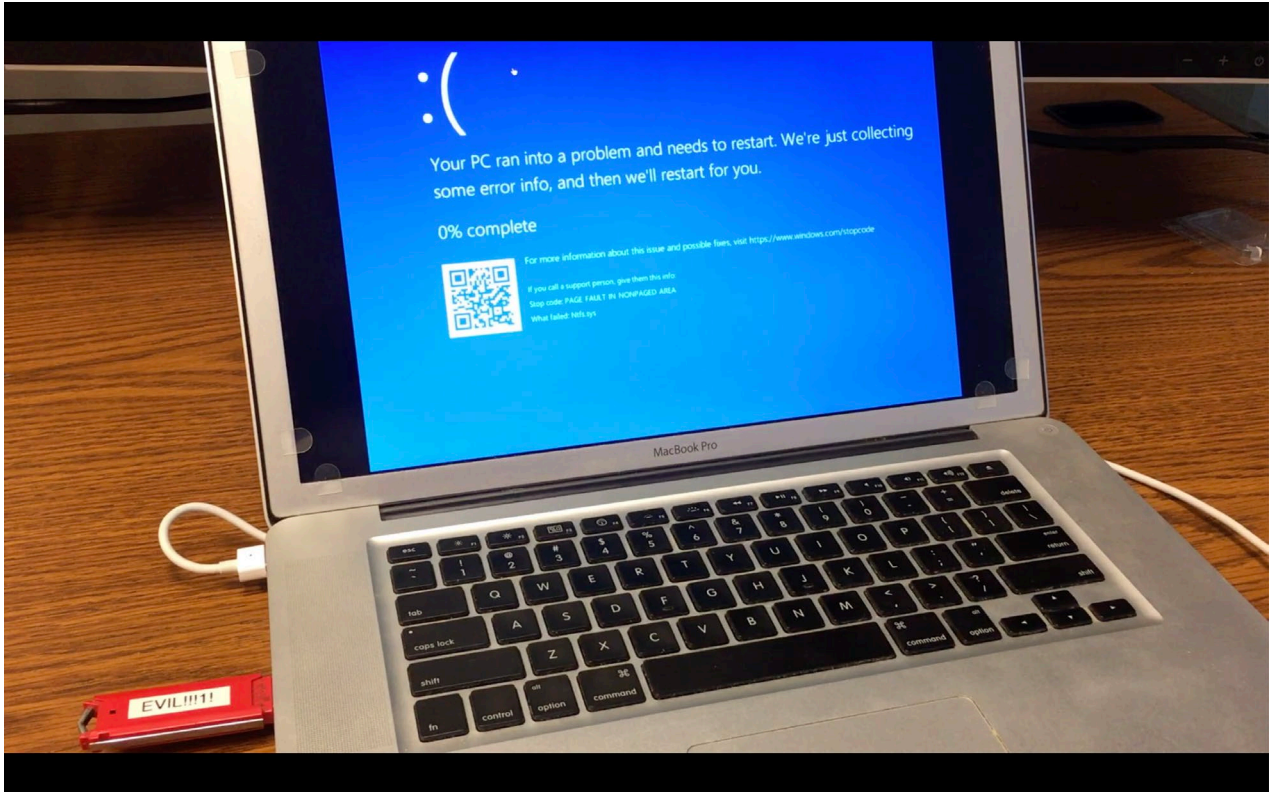
Death by Thumb Drive

Corrupt File System Attack Vectors



Do We Need Physical Access?

Watch the video here.



What About the Macs?

Attacker renames the dd image to .dmg (Apple Disk Image)

Safari auto-downloads DMG files

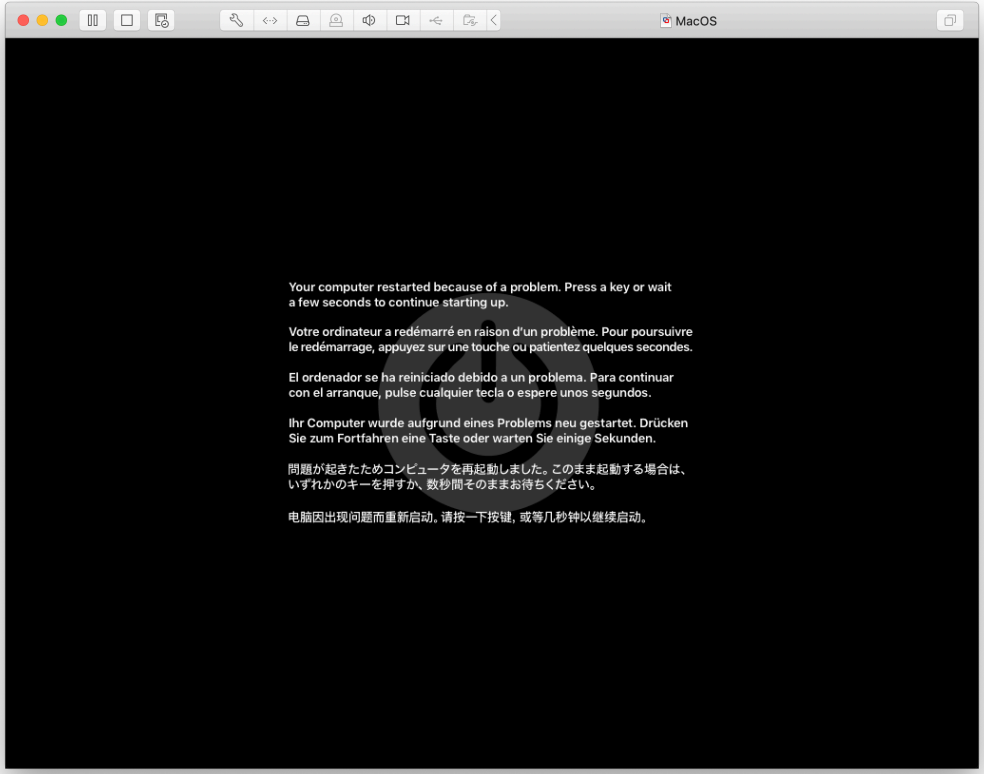
User double-clicks DMG

What About the Macs?

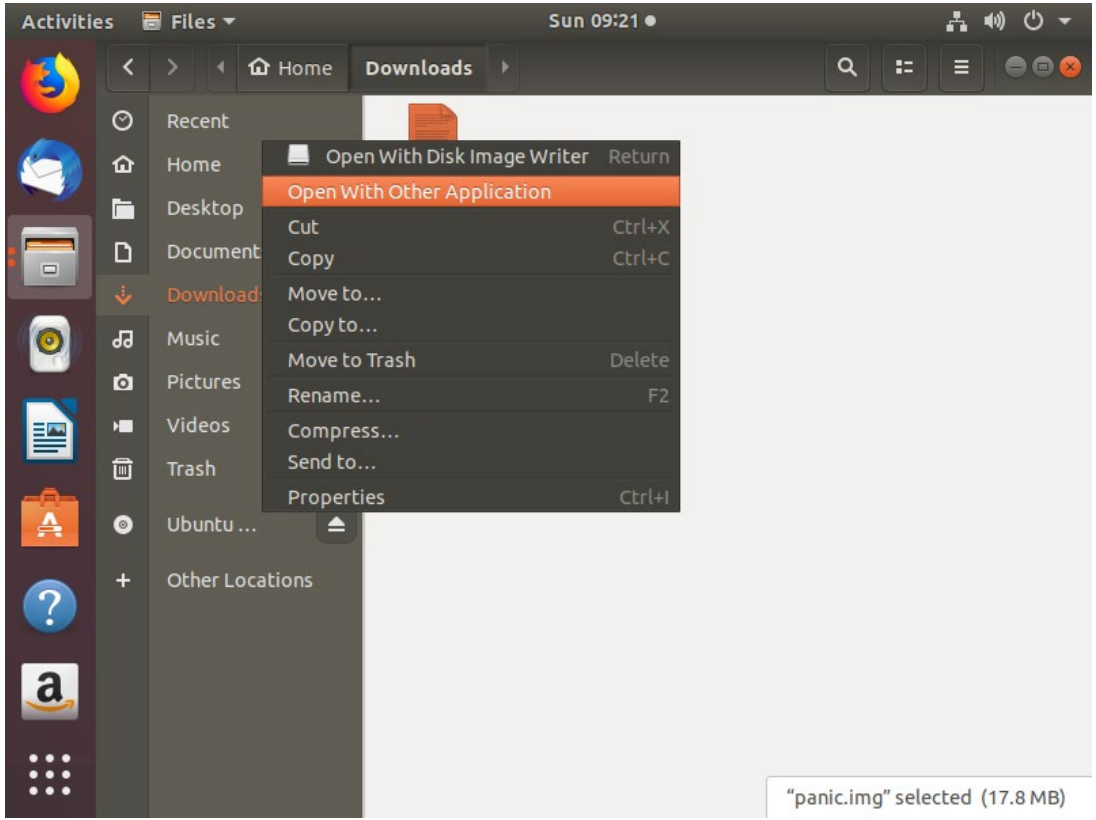
Attacker renames the dd image to .dmg (Apple Disk Image)

Safari auto-downloads DMG files

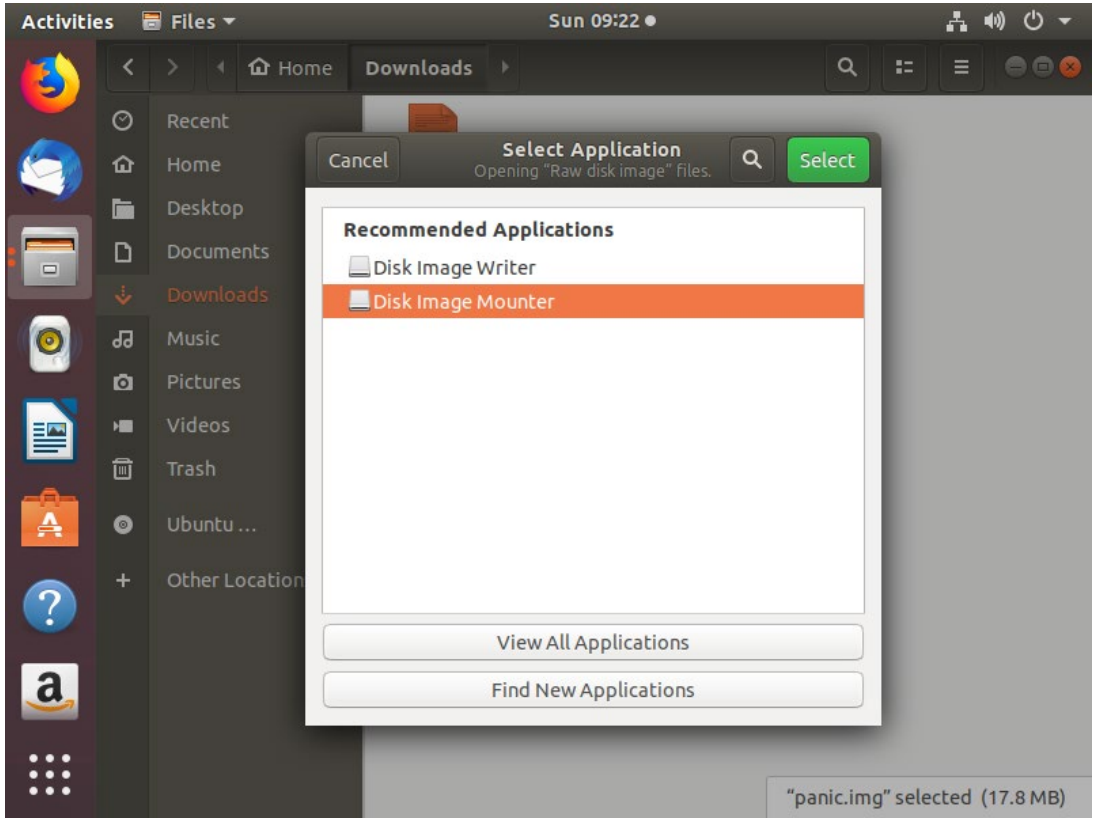
User double-clicks DMG



Ubuntu Linux?



Manual Interaction Required



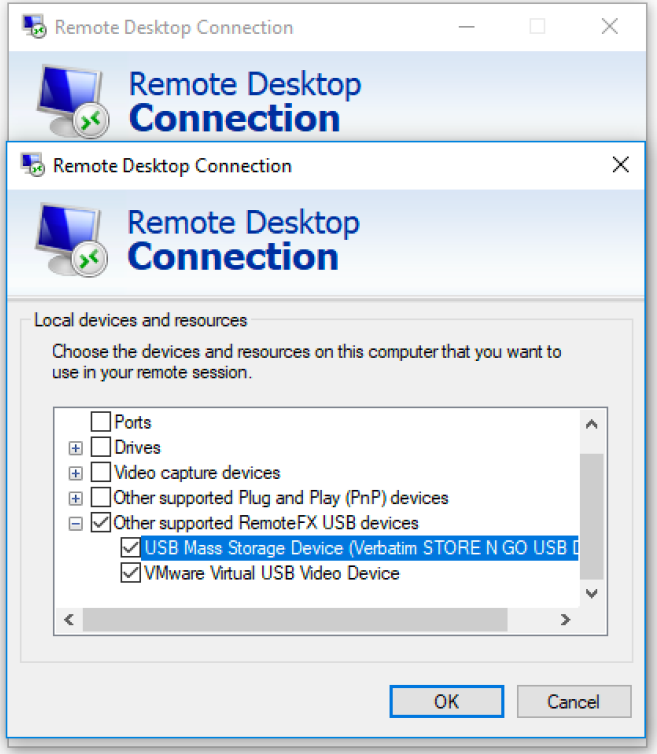
And Then…

```
[ 1244.616676] BUG: unable to handle kernel NULL pointer dereference at 0000000000000018
[ 1244.624092] PGD 0 P4D 0
[ 1244.625489] Oops: 0000 [#1] SMP NOPTI
[ 1244.632268] CPU: 0 PID: 10637 Comm: mount Kdump: loaded Not tainted 4.20.13 #3
[ 1244.639026] Hardware name: VMware, Inc. VMware7,1/440BX Desktop Reference Platform, BIOS VMW71.00V.0.B64.1508272355 08/27/201
5
[ 1244.648919] RIP: 0010:journal_init+0x109b/0x1670 [reiserfs]
[ 1244.653004] Code: 8b 85 50 ff ff ff 42 8b 74 b0 0c 48 8b bb d0 00 00 00 8b 53 18 b9 08 00 00 00 e8 10 31 2c dd 49 89 45 00 48
 8b 8b d8 03 00 00 <4c> 8b 68 18 48 8b 79 08 8b 07 49 39 c5 0f 87 ce 03 00 00 48 8b 41
[ 1244.666585] RSP: 0018:ffffc90002a3fbb0 EFLAGS: 00010286
[ 1244.669280] RAX: 0000000000000000 RBX: ffff888027cb2000 RCX: ffff888027cc0a00
[ 1244.679365] RDX: 0000000000000000 RSI: ffff88807a01fb80 RDI: ffffea00008129c0
[ 1244.683459] RBP: ffffc90002a3fcb8 R08: 0000000000000000 R09: ffff88807a501c80
[ 1244.690394] R10: 0000000000000000 R11: 000015ffff7ed63f R12: 000000000000000c
[ 1244.697517] R13: ffff888027c74c60 R14: 000000000000000c R15: ffff888027c74460
[ 1244.703969] FS:  00007f18f5a14080(0000) GS:ffff88807aa00000(0000) knlGS:0000000000000000
[ 1244.711510] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[ 1244.717399] CR2: 0000000000000018 CR3: 000000007e0f8000 CR4: 00000000000406f0
[ 1244.724164] Call Trace:
[ 1244.726094]  reiserfs_fill_super+0x4c2/0xca0 [reiserfs]
[ 1244.737688]  ? snprintf+0x45/0x70
[ 1244.740106]  mount_bdev+0x17f/0x1b0
[ 1244.751256]  ? finish_unfinished+0x680/0x680 [reiserfs]
[ 1244.754699]  get_super_block+0x15/0x20 [reiserfs]
[ 1244.756121]  mount_fs+0x37/0x150
[ 1244.766518]  vfs_kern_mount.part.26+0x5d/0x110
[ 1244.769301]  do_mount+0x5ed/0xce0
[ 1244.775523]  ? memdup_user+0x4f/0x80
[ 1244.782106]  ksys_mount+0x98/0xe0
[ 1244.783564]  __x64_sys_mount+0x25/0x30
[ 1244.792971]  do_syscall_64+0x5a/0x120
[ 1244.797763]  entry_SYSCALL_64_after_hwframe+0x44/0xa9
[ 1244.810625] RIP: 0033:0x7f18f52c23ca
[ 1244.812554] Code: 48 8b 0d c1 8a 2c 00 f7 d8 64 89 01 48 83 c8 ff c3 66 2e 0f 1f 84 00 00 00 00 00 0f 1f 44 00 00 49 89 ca b8
 a5 00 00 00 0f 05 <48> 3d 01 f0 ff ff 73 01 c3 48 8b 0d 8e 8a 2c 00 f7 d8 64 89 01 48
[ 1244.826709] RSP: 002b:00007ffec2482008 EFLAGS: 00000202 ORIG_RAX: 00000000000000a5
[ 1244.828020] RAX: ffffffffffffffda RBX: 0000556b2d142a40 RCX: 00007f18f52c23ca
[ 1244.839582] RDX: 0000556b2d14cb80 RSI: 0000556b2d142c40 RDI: 0000556b2d142c20
[ 1244.843272] RBP: 0000000000000000 R08: 0000000000000000 R09: 00007f18f530e1b0
[ 1244.844546] R10: 00000000c0ed0000 R11: 0000000000000202 R12: 0000556b2d142c20
[ 1244.856746] R13: 0000556b2d14cb80 R14: 0000000000000000 R15: 00007f18f57ea8a4
[ 1244.857983] Modules linked in: reiserfs hfs f2fs ntfs nilfs2 minix hfsplus xfs nls_utf8 isofs ufs nfsv3 nfs_acl rpcsec_gss_kr
b5 auth_rpcgss nfsv4 nfs lockd grace fscache nls_iso8859_1 vmw_balloon crct10dif_pclmul crc32_pclmul ghash_clmulni_intel aesni_i
ntel aes_x86_64 crypto_simd cryptd glue_helper serio_raw vmw_vmci sunrpc sch_fq_codel ip_tables x_tables autofs4 btrfs xor zstd_
compress raid6_pq libcrc32c drm_kms_helper syscopyarea sysfillrect sysimgblt fb_sys_fops ttm drm psmouse e1000 i2c_piix4 i2c_cor
e ahci vmw_pvscsi libahci pata_acpi floppy
[ 1244.905050] CR2: 0000000000000018
```

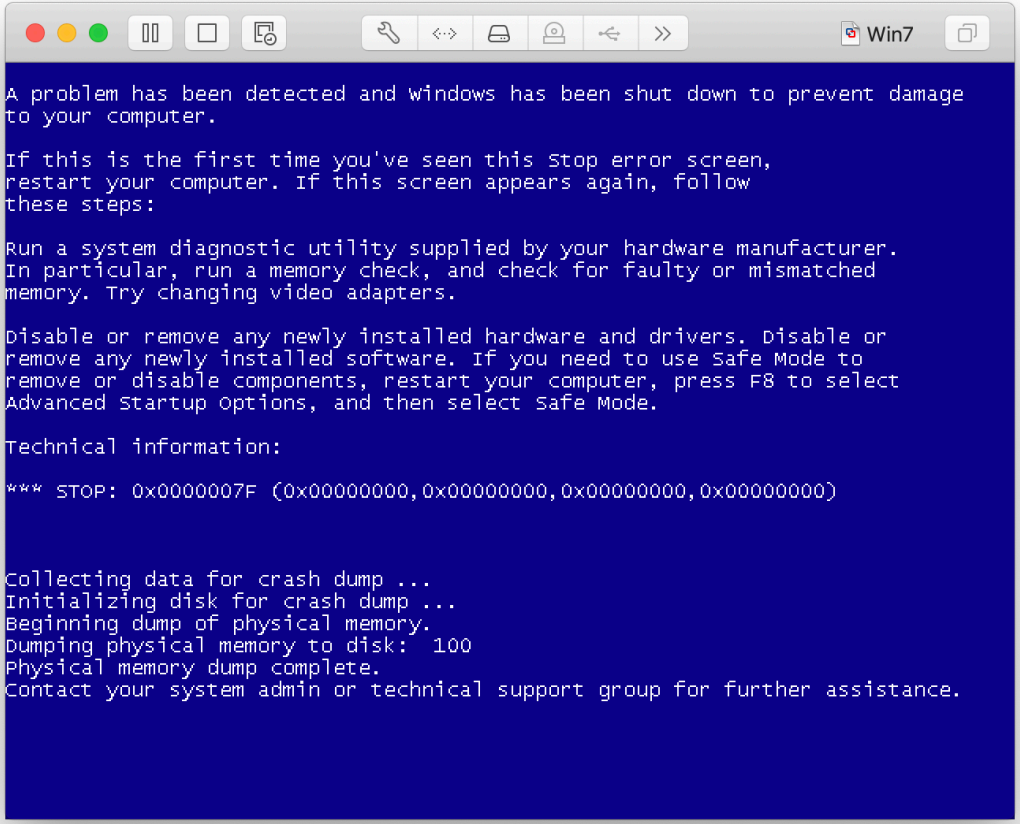
Windows RDP RemoteFX

RemoteFX allows USB Device pass-through

- Optional RDP feature
- Only for authenticated users



After Connecting USB via RemoteFX

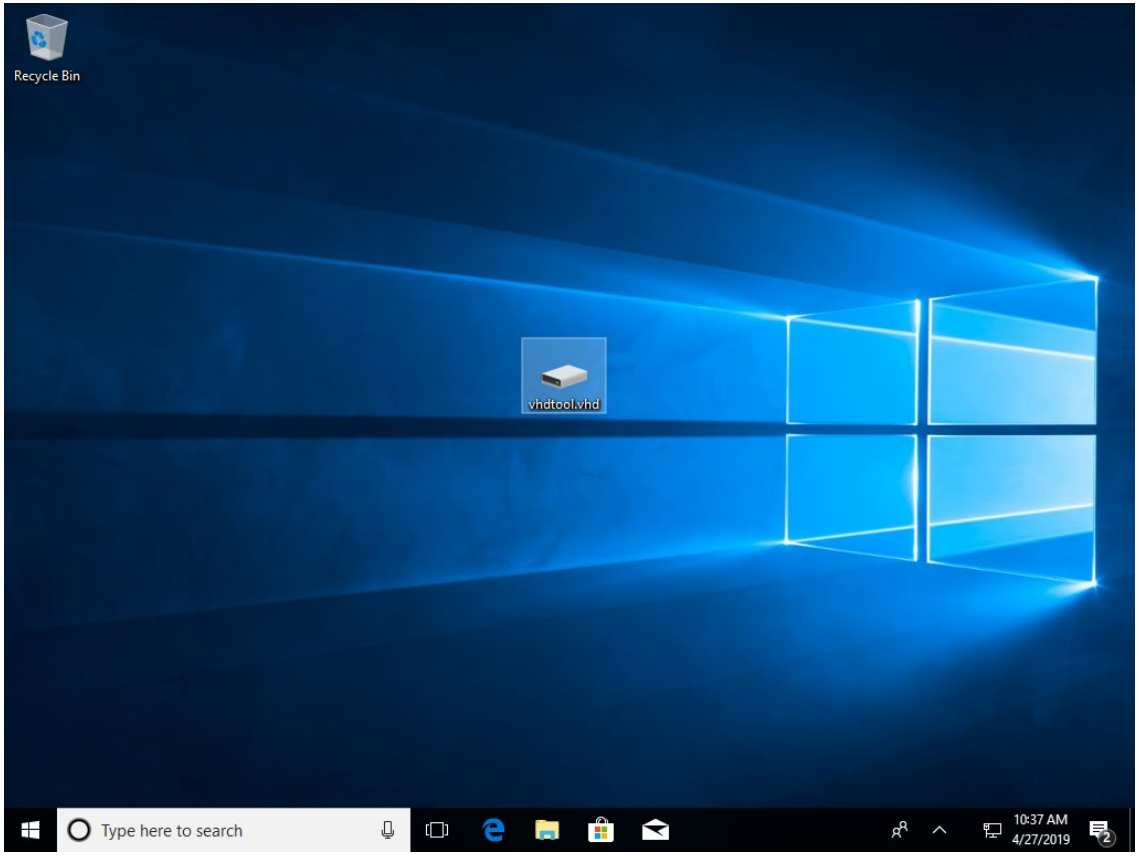


vhdtool

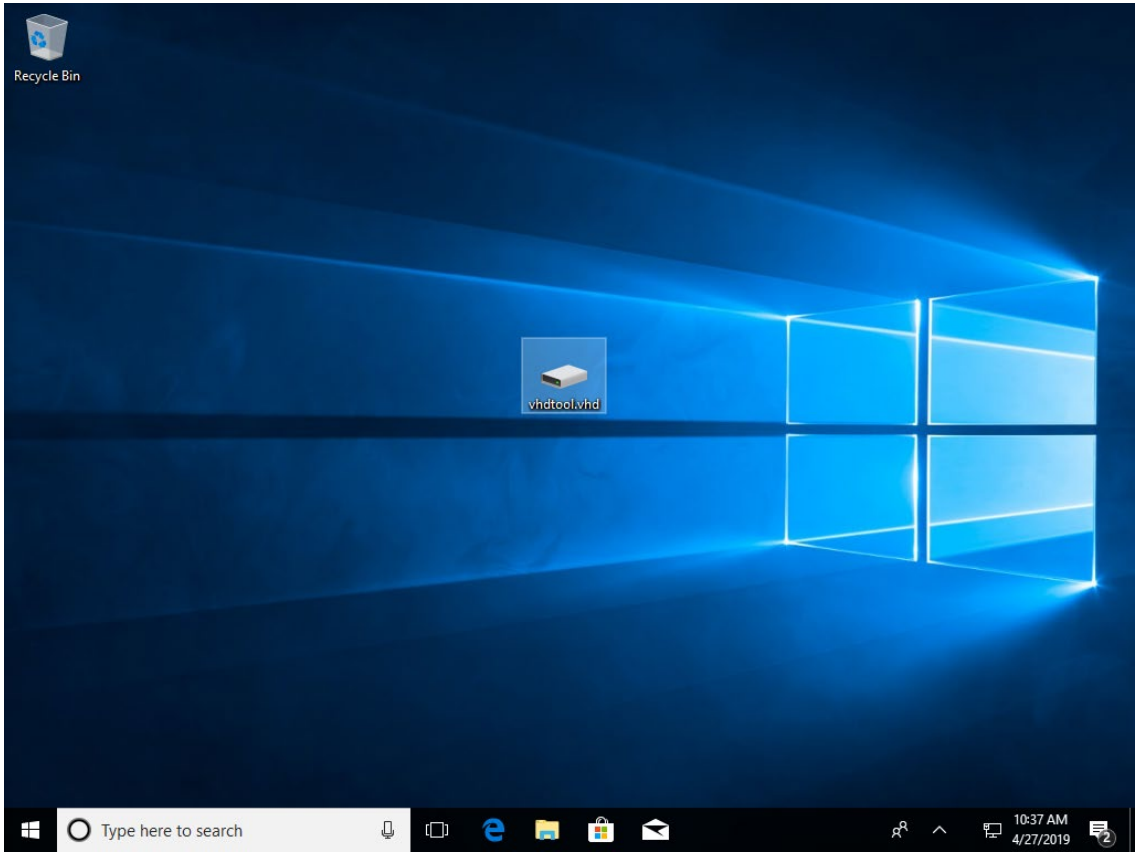
[Read the code on GitHub here]

```
VHDtool
==========

A tool to examine and manipulate VHD images. Initially
meant as a way to test dynamic VHD support in my
Linux kernel loop VHD parser support. Images mount
in Win7.

Why would you use this instead of qemu-img? VHDtool
lets you tweak more parameters and create funky
VHDs (and will support differencing disks soon too!).
```

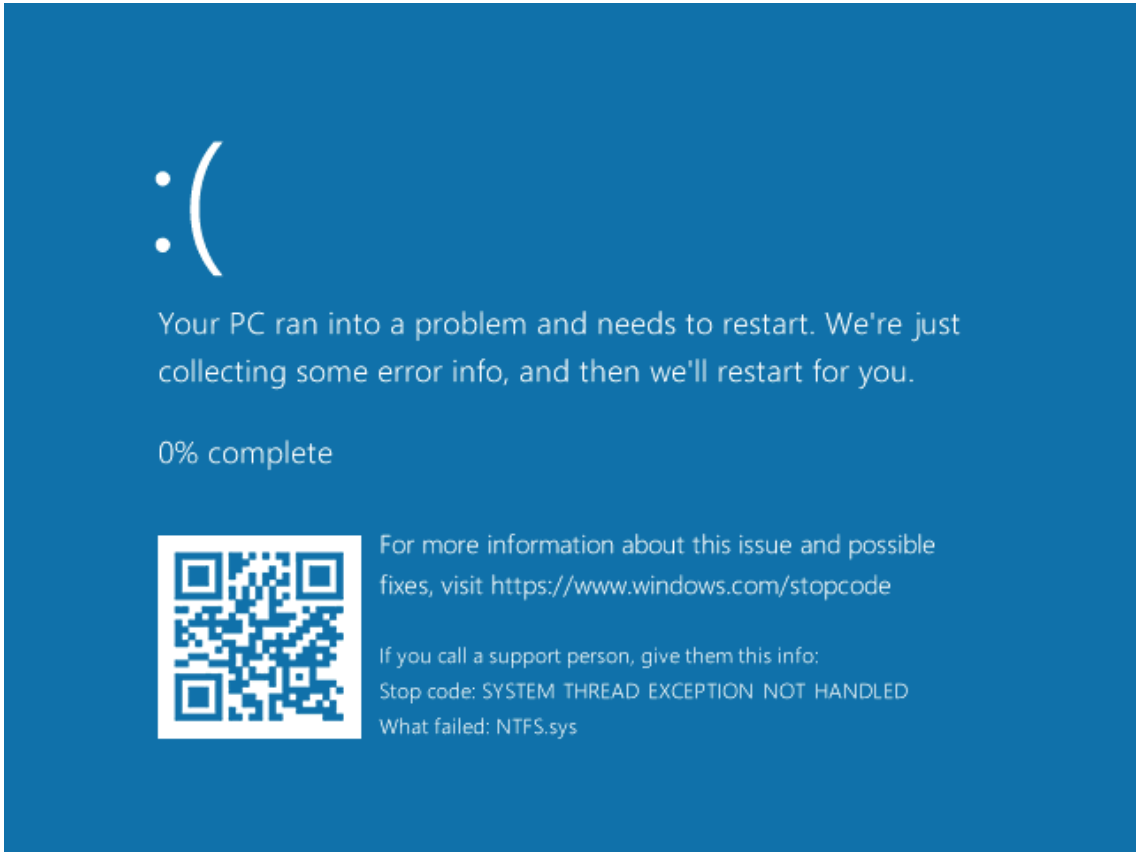
So We Downloaded a VHD…



And Double-clicked it…



And Double-clicked it…



Are Your Security Products Scanning VHD(X)?

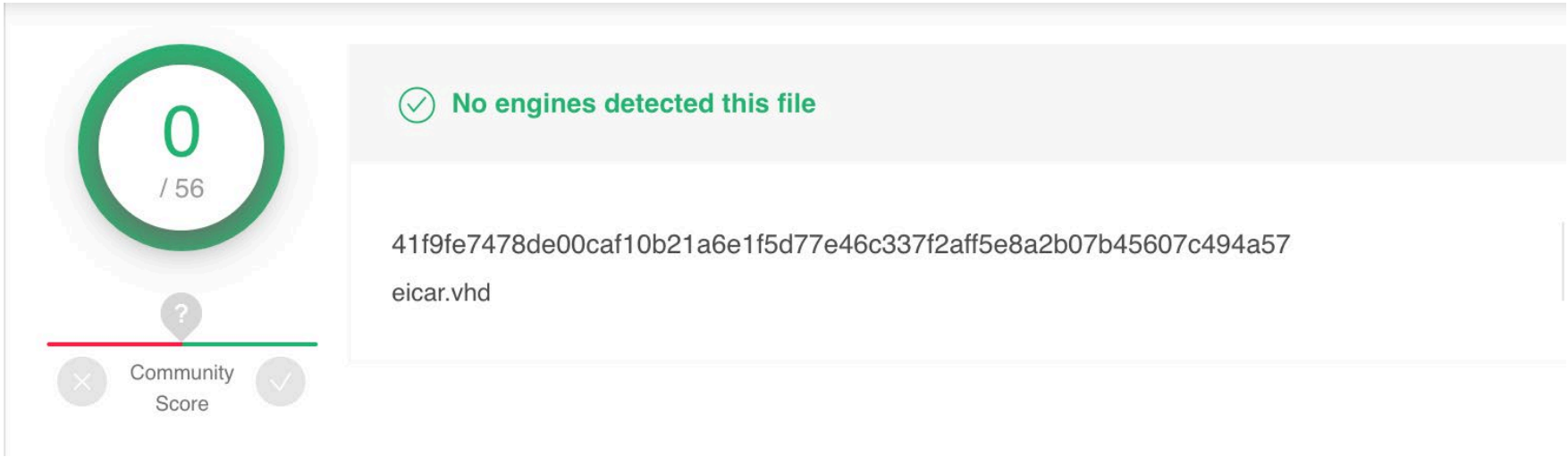
- **Yes**

  Are you prepared for your security products to crash with malformed VHD(X) files?

- **No**

  Are you prepared for clickable files to reach your endpoints without any visibility from your security products?

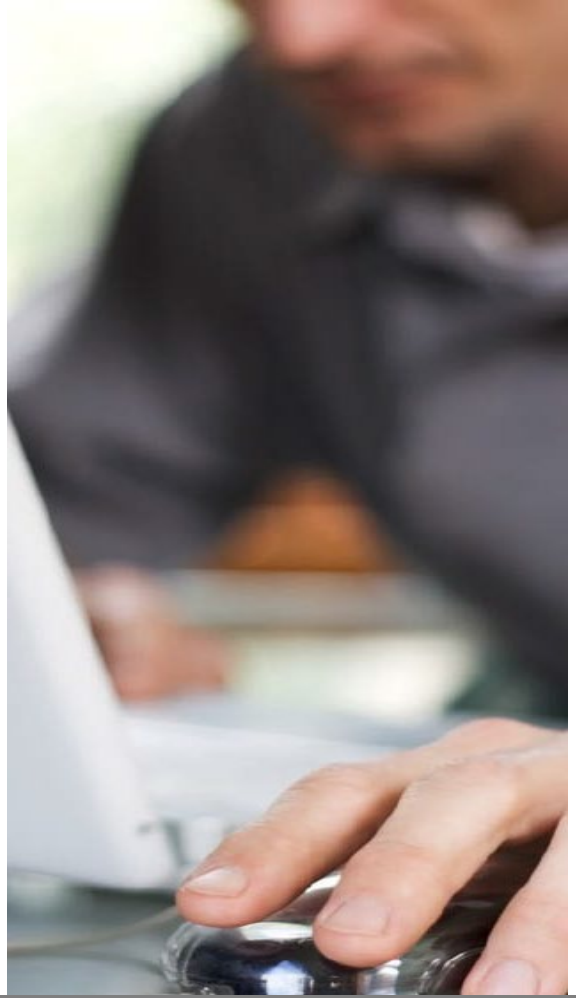
No, Your Security Products Don’t Scan VHD



[Read more on the SEI site here.]

Death by Thumb Drive

Recommendations



Recommendations

Unless you're **certain** your OS does not auto-mount filesystems, **do not** plug unknown USB devices into your computer.

Hint: macOS, Ubuntu, and Windows all auto-mount drives

Even if you're certain that your OS does not auto-mount filesystems, **do not** plug unknown USB devices into your computer!

Block **VHD** and **VHDX** at email and other gateways.

If you have **RemoteFX** enabled, confirm that you actually need it.

What Does This Do?

Contact Information

Presenter / Point of Contact

Will Dormann

Senior Vulnerability Analyst

Email: wd@cert.org

Twitter: @wdormann

CERT Coordination Center

Email: cert@cert.org

Twitter: @certcc

Blog: http://insights.sei.cmu.edu/cert/