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

Why Fuzz File Systems?

Background

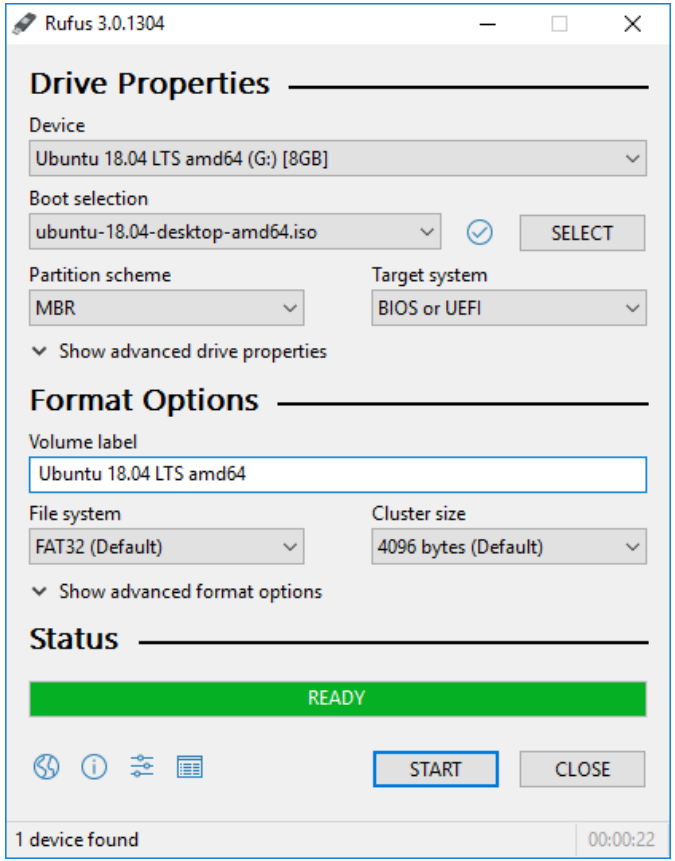
FreeNAS®

FreeBSD-based ZFS file server distribution.

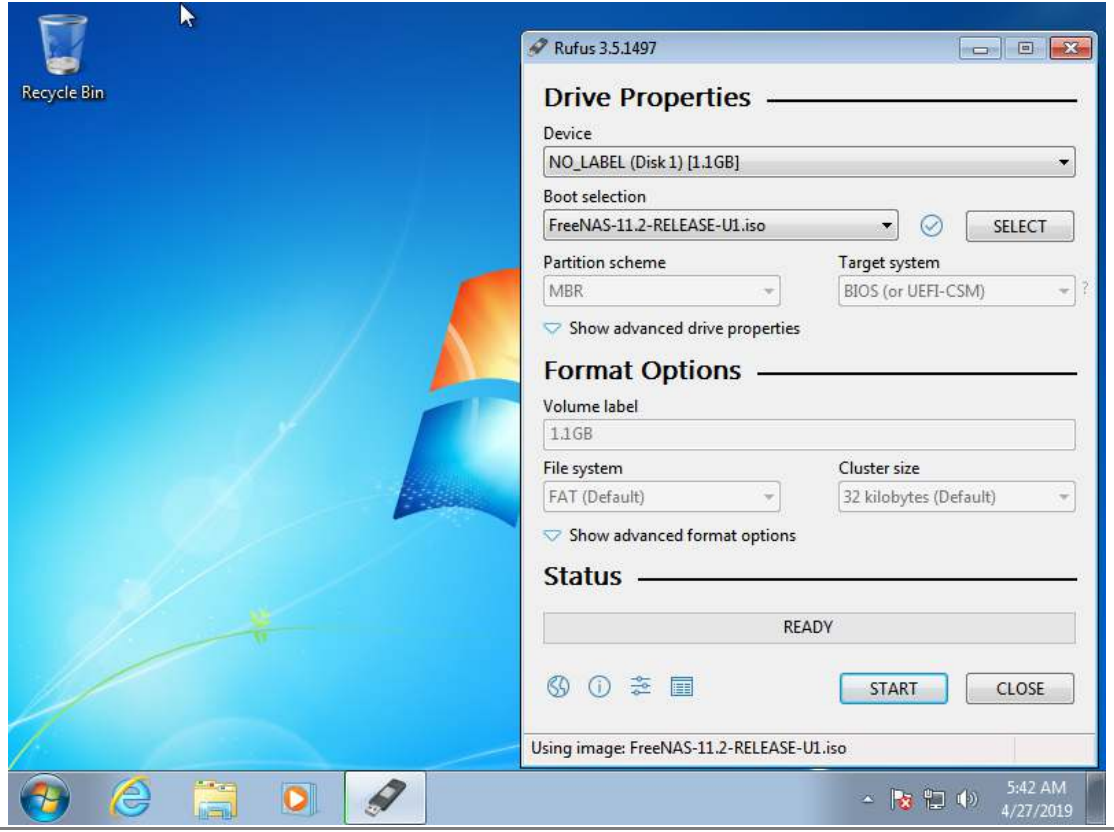
Background

Rufus

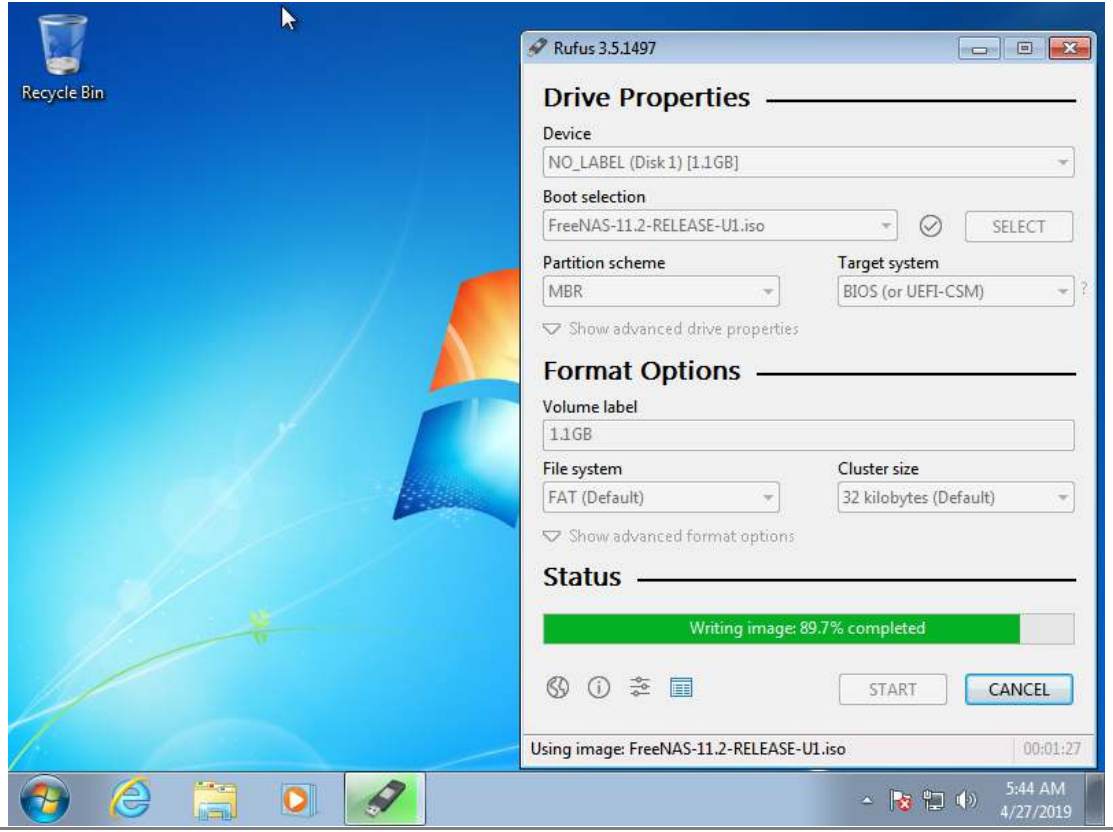
Allows creating bootable USB drives from ISO images.



Creating a FreeNAS 11.2 USB Drive



Creating a FreeNAS 11.2 USB Drive



Creating a FreeNAS 11.2 USB Drive

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to be sure you have adequate disk space. If a driver is
identified in the Stop message, disable the driver or check
with the manufacturer for driver updates. Try changing video
adapters.

Check with your hardware vendor for any BIOS updates. Disable
BIOS memory options such as caching or shadowing. If you need
to use Safe Mode to remove or disable components, restart your
computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x0000007E (0xFFFFFFFFC0000094,0xFFFFF80002AC45F9,0xFFFFF88002FD4FC8,0
xFFFFF88002FD4830)



Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  55
```

Vulnerability Discovery for Everyone

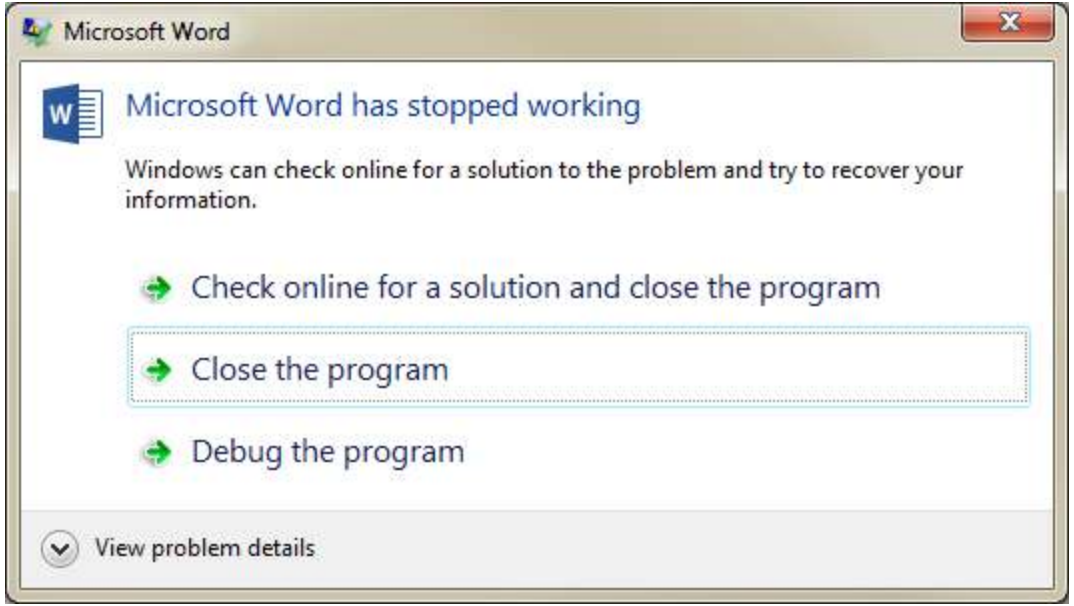
How to discover vulnerabilities:

1. Use systems
2. Notice anomalies
3. Investigate anomalies

Death by Thumb Drive

CERT BFF Background

Mutational Fuzzing



The CERT BFF

* It's not you, it's me

The CERT Basic Fuzzing Framework

* It's not you, it's me

How BFF Works

1. Pick with a seed file
2. Mangle the file
3. Launch target application
4. Look for crashes
5. Analyze crash
6. Repeat

Do this blindly, but as as intelligently as possible

https://vuls.cert.org/confluence/display/tools/CERT+BFF+-+Basic+Fuzzing+Framework

Checking Results with tools/drillresults.py

```
fuzz@UbuFuzz64: ~/bff
--- Interesting crashes ---


3272c66a161219e445ed456df99a2cc1 - Exploitability rank: 10
Fuzzed file: ../results/outsidein852/crashers/3272c66a161219e445ed456df99a2cc1/sf_e34596126cc8b1f
169af2c6d950bc68d-754-minimized.DB
exception 0: ReturnAv accessing 0x32c0000000000000   *** Byte pattern is in fuzzed file! ***
ret
Code executing in: /home/fuzz/in/oi/redist/libvs_pdx.so

47a81315c0fec4e70da279063230c71c - Exploitability rank: 10
Fuzzed file: ../results/outsidein852/crashers/47a81315c0fec4e70da279063230c71c/sf_bd0d5dbc9dc4124
cde7135c718a83488-18132-minimized.doc
exception 0: ReturnAv accessing 0x02c0000f000102bf   *** Byte pattern is in fuzzed file! ***
ret
Code executing in: /home/fuzz/in/oi/redist/libvs_eshr.so

6bb821b56f0d688a0ebfa27c10347de7 - Exploitability rank: 10
Fuzzed file: ../results/outsidein852/crashers/6bb821b56f0d688a0ebfa27c10347de7/sf_9630c50c60dd86a
1524c148cfa0b8d8b-8481-minimized.wk4
exception 0: ReturnAv accessing 0x00022a310018001b   *** Byte pattern is in fuzzed file! ***
ret
Code executing in: /home/fuzz/in/oi/redist/libvs_wk4.so

:
```

Confirming Control of RET

edb - /home/fuzz/in/oi/sdk/demo/exsimple [1812]

File View Debug Plugins Options Help

No Analysis Found For This Region

```
00002aaa:ae64ff58 c3                    ret
00002aaa:ae64ff59 0f 1f 80 00 00 00 00  nop dword ptr [rax]
00002aaa:ae64ff60 c7 84 24 6c 01 00 00 0…  mov dword ptr [rsp+364], 3
00002aaa:ae64ff6b 66 c7 85 74 09 00 00 0…  mov word ptr [rbp+2420], 1
00002aaa:ae64ff74 e9 cf fe ff ff        jmp 0x00002aaaae64fe48
00002aaa:ae64ff79 0f 1f 80 00 00 00 00  nop dword ptr [rax]
00002aaa:ae64ff80 66 41 83 ff 0b        cmp r15w, 11
00002aaa:ae64ff85 0f 8f fb 06 00 00     jnle 0x00002aaaae650686
00002aaa:ae64ff8b 0f b6 54 24 24        movzx edx, byte ptr [rsp+36]
00002aaa:ae64ff90 c6 85 06 1e 00 00 00  mov byte ptr [rbp+7686], 0
00002aaa:ae64ff97 8d 44 92 22           lea eax, [rdx+rdx*4+34]
00002aaa:ae64ff9b 88 85 05 1e 00 00     mov byte ptr [rbp+7685], al
00002aaa:ae64ffa1 e9 fb fe ff ff        jmp 0x00002aaaae64fea1
00002aaa:ae64ffa6 66 41 83 ff 0b        cmp r15w, 11
00002aaa:ae64ffab 0f 8e f7 06 00 00     jle 0x00002aaaae6506a8
00002aaa:ae64ffb1 c6 85 05 1e 00 00 48  mov byte ptr [rbp+7685], 72
00002aaa:ae64ffb8 0f b6 54 24 24        movzx edx, byte ptr [rsp+36]
00002aaa:ae64ffbd 80 bd 02 1e 00 00 00  cmp byte ptr [rbp+7682], 0
00002aaa:ae64ffc4 c6 85 07 1e 00 00 00  mov byte ptr [rbp+7687], 0
00002aaa:ae64ffcb 8d 44 92 21           lea eax, [rdx+rdx*4+33]
```

return to 4142434445464748

Registers

```
General Purpose
RAX: 0000000000000003
RBX: 8100000000150000
RCX: 0000000000000000
RDX: 0000000000000000
RBP: 810f0001ff010001
RSP: 00007fffffffdc558 ASCII "H…
RSI: 00000000ffffffff
```

Bookmarks

Address	Comment

Add Del Clear

Data Dump

0000000000400000-0000000000417000

```
00000000:00400000 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00
00000000:00400010 02 00 3e 00 01 00 00 00 e0 24 40 00 00 00
00000000:00400020 40 00 00 00 00 00 00 00 78 ba 01 00 00 00
00000000:00400030 00 00 00 00 40 00 38 00 08 00 40 00 1b 00
00000000:00400040 06 00 00 00 05 00 00 00 40 00 00 00 00 00
00000000:00400050 40 00 40 00 00 00 00 00 40 00 40 00 00 00
00000000:00400060 c0 01 00 00 00 00 00 00 c0 01 00 00 00 00
00000000:00400070 08 00 00 00 00 00 00 00 03 00 00 00 04 00
```

Stack

```
00007fff:fffdc558 4142434445464748 HGFEDCBA
00007fff:fffdc560 001b002100000000 ....!...
00007fff:fffdc568 000000022a33000a  .3*....
00007fff:fffdc570 0028001b00210001 ..!...(.
00007fff:fffdc578 0021000000022a34 4*....!.
00007fff:fffdc580 0021002100210021 !.!.!.!.
00007fff:fffdc588 0021002100210021 !.!.!.!.
00007fff:fffdc590 002e002100230023 #.#.!...
00007fff:fffdc598 0021002100210021 !.!.!.!.
00007fff:fffdc5a0 00022a350068001b ..h.5*..
00007fff:fffdc5a8 0000a80000ff0000 ..□..□..
00007fff:fffdc5b0 0100a80501ff02ff □.□..□..
```

paused

Death by Thumb Drive

BFF Enhancements

BFF Enhancements

https://github.com/CERTCC/certfuzz/pull/24 by antnks

1. Copy fuzzed file to a fixed location
2. Run a program for each iteration

Copy fuzzed file to a fixed location

Typical fuzzed file location:

```
/home/fuzz/fuzzing/campaign_UwykZc/iteration_2soD_J/BFF_testcase_yQf1fz/sf_e7795bfdec1e75189fa96cdfcc915c17-1383.img
```

```
target:
    program: ~/convert
    cmdline_template: $PROGRAM $SEEDFILE /dev/null
    copyfuzzedto: /tmp/fuzzedfile
```

After bff.yaml modification, a copy of each iteration is placed in:

```
/tmp/fuzzedfile
```

Run a program for each iteration

After each file is mutated, you can run an arbitrary program.

```
target:
    program: ~/convert
    cmdline_template: $PROGRAM $SEEDFILE /dev/null
    copyfuzzedto: /tmp/fuzzedfile
    postprocessfuzzed: /usr/local/bin/postprocess /tmp/fuzzedfile
```

After bff.yaml modification, the user-specified program is executed for each iteration:

```
/usr/local/bin/postprocess /tmp/fuzzedfile
```

Putting Things Together…

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
```

Death by Thumb Drive

Fuzzing Filesystems with BFF

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
```

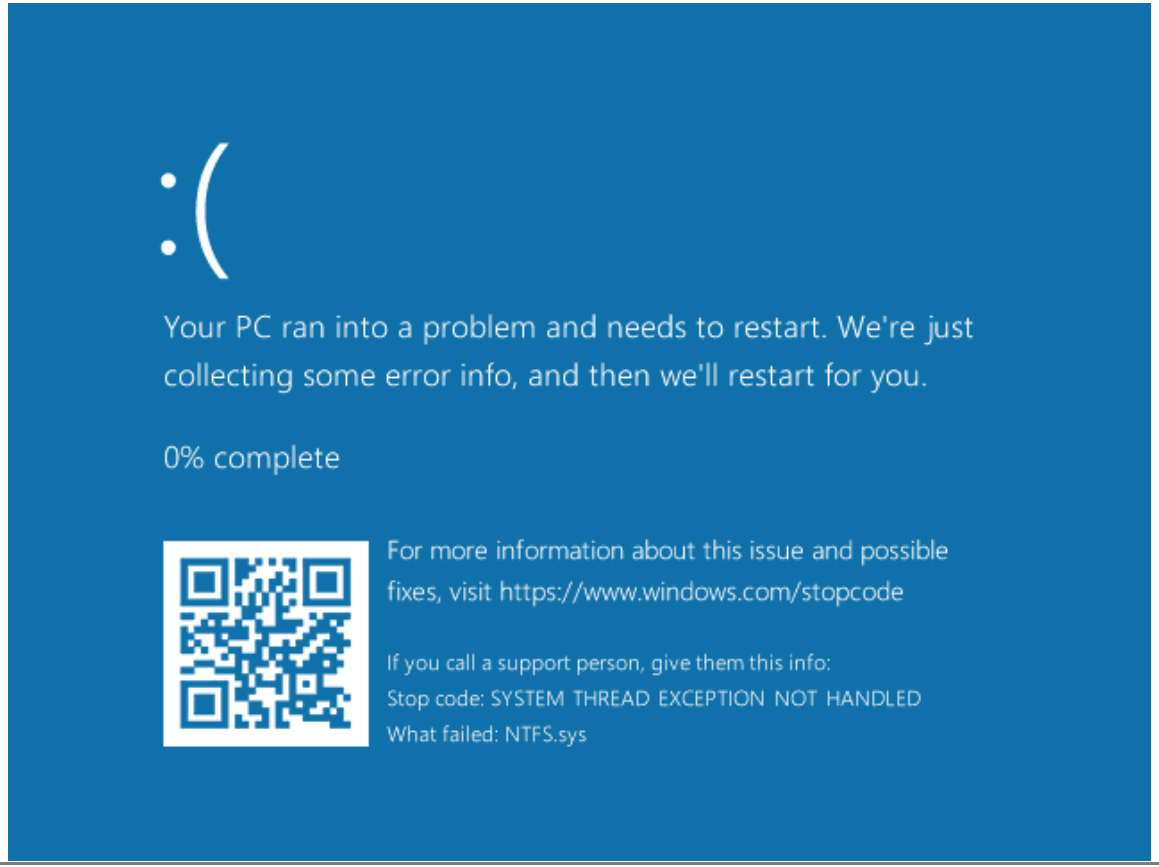
Eventually

```
[ 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
```

Or on Windows

```
dd if=/cygdrive/c/tmp/fuzzed.bin of=/dev/sdc bs=10M
diskpart /s c:\users\fuzz\rescan.txt
explorer e:\
c:\cygwin\bin\find.exe /cygdrive/e
pskill explorer.exe
```

Eventually



Or on macOS

```
#!/bin/bash
rootdisk=`mount | grep "on / " | awk '{print $1}'`
if [ "$rootdisk" == "/dev/disk0s2" ]; then
  extdisk=disk1
fi
diskutil unmount force /dev/disk2s1
diskutil unmountDisk force /dev/${extdisk}
dd if=/Users/test/fs.bin of=/dev/r${extdisk} bs=1m count=10
diskutil unmountDisk force /dev/disk2s1
diskutil unmountDisk force /dev/${extdisk}
```

Eventually

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

https://yarchive.net/comp/linux/debuggers.html

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:
https://help.ubuntu.com/lts/serverguide/kernel-crash-dump.html.en

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"
```

Collecting Linux Core Dumps

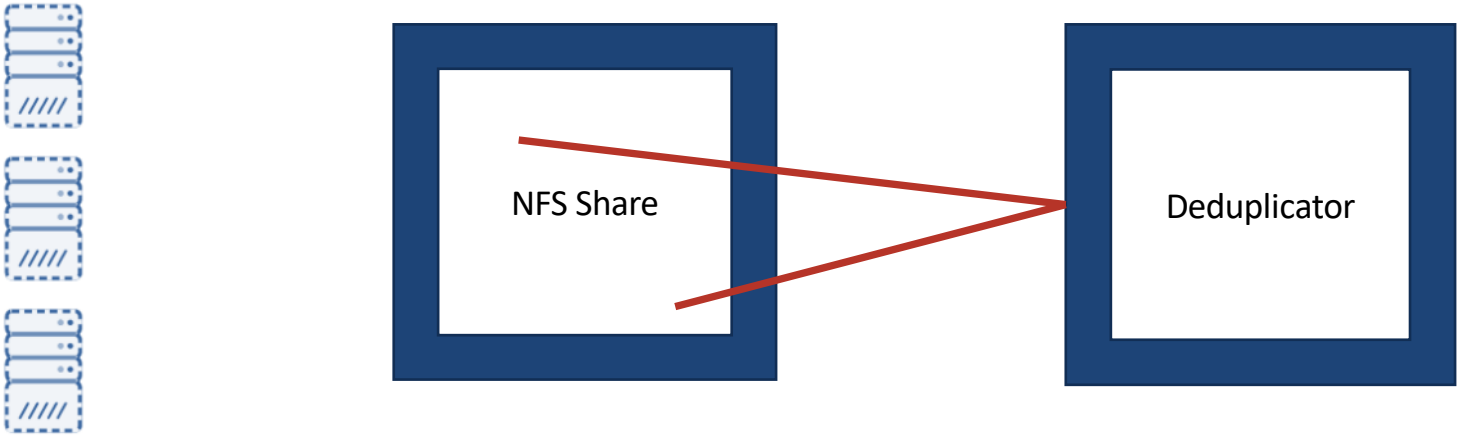
Edit `/etc/default/kdump-tools`

```
NFS="NFS_SERVER:/exported/share"
NFS_TIMEO="600"
NFS_RETRANS="3"
```

Edit `/usr/sbin/kdump-config`

```
log_action_msg "Getting fuzzed filesystem..."
dd if=/dev/sdb of=$KDUMP_STAMPDIR/panic.bin bs=1M
```

Automation of Collection



Come back later for results

Death By Thumb Drive

Investigating Crashes

Windows

VirtualKD

Virtual Machine monitor

PID	VM type	Uptime	CPU	Pipe name	Packets	Resets	OS	Debugger
540	VMWare x64	00:02:42	0%	kd_VMXPPRO	251/252	2	yes	yes
2864	VirtualBox x64	00:00:53	24%	kd_VMSRV	577/566	18	yes	yes

Bytes received	203K
Bytes send	44K
Packets received	577
Packets sent	566
IN packet rate	0/s
OUT packet rate	0/s
Reset count	18
Send rate	0/s
Receive rate	0/s
Max. send rate	50K/s
Max. recv. rate	373K/s
CPU usage	24%
Avg. CPU usage	12%

Debug message level: Patcher debug messages

Clear log

Start debugger automatically

Stop debugger automatically

DbgBreakPoint() on start

WINDBG.EXE

KD.EXE

Custom: cmd.exe /c "$(toolspath)\test.cmd" $(pipename)

TraceAssist params...

Debugger path...

Run debugger

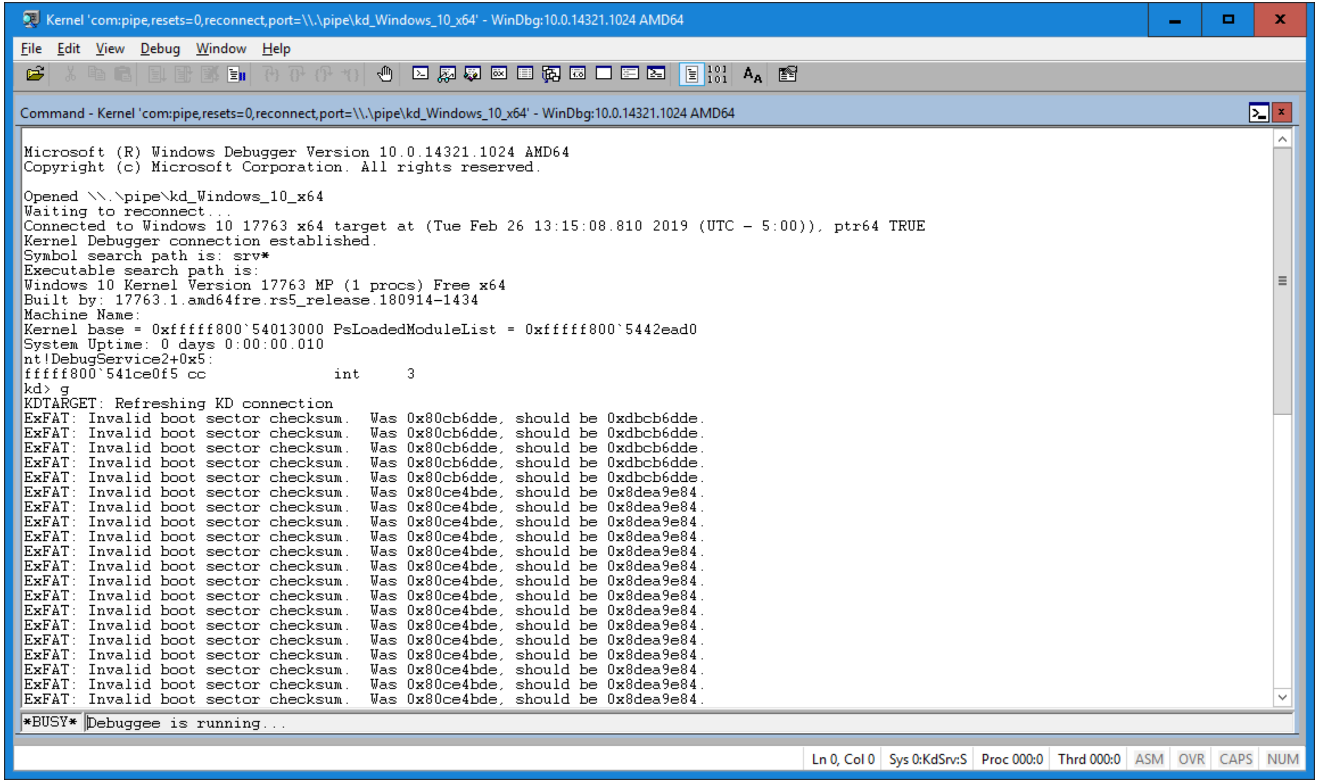
Log all sent and received packets to kd_<VMNAME>.html files (useful for analyzing debugger protocol)

Unpatch process

Close

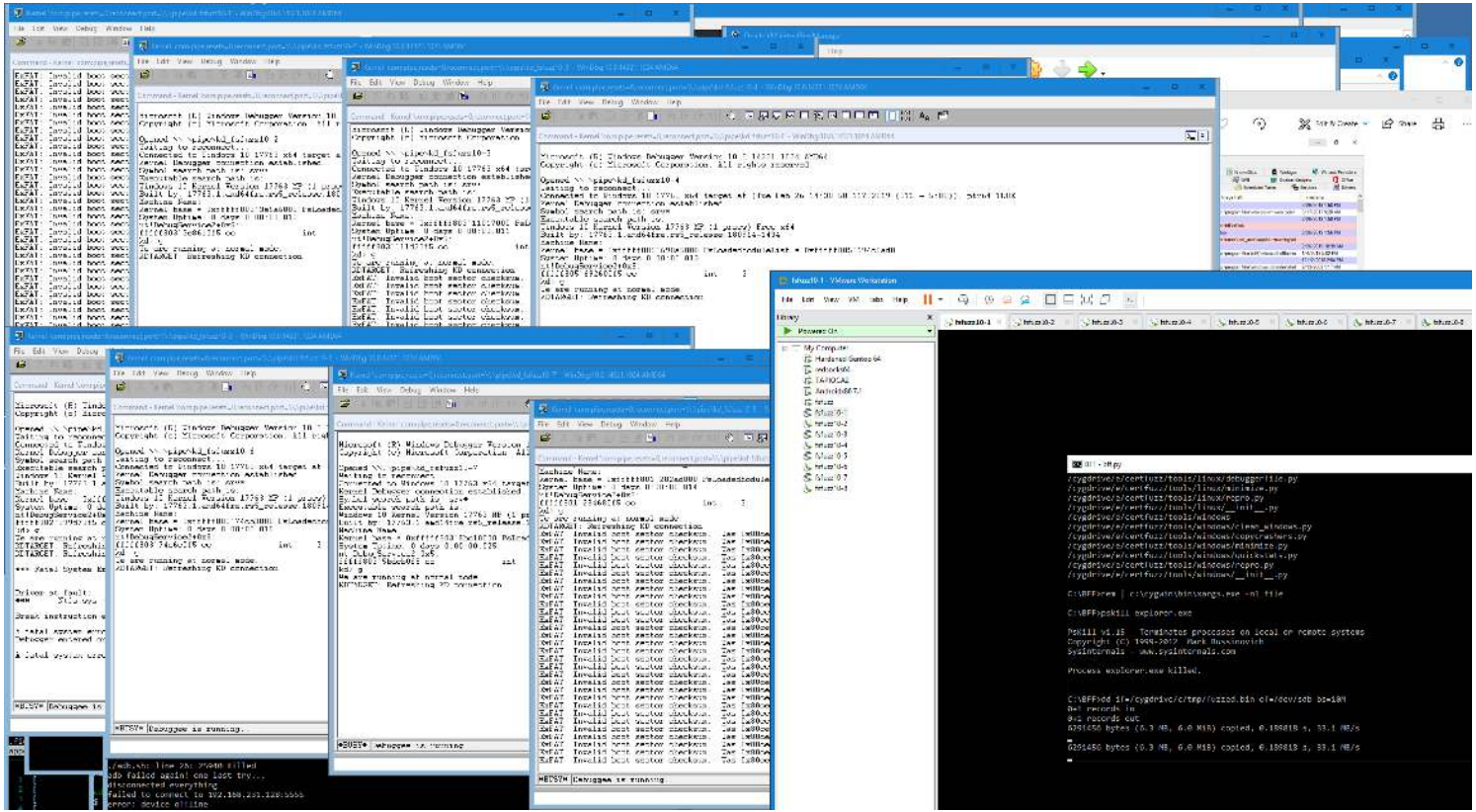
http://sysprogs.com/legacy/virtualkd/

VirtualKD in Action



```
Microsoft (R) Windows Debugger Version 10.0.14321.1024 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\kd_Windows_10_x64
Waiting to reconnect...
Connected to Windows 10 17763 x64 target at (Tue Feb 26 13:15:08.810 2019 (UTC - 5:00)), ptr64 TRUE
Kernel Debugger connection established.
Symbol search path is: srv*
Executable search path is:
Windows 10 Kernel Version 17763 MP (1 procs) Free x64
Built by: 17763.1.amd64fre.rs5_release.180914-1434
Machine Name:
Kernel base = 0xfffff800`54013000 PsLoadedModuleList = 0xfffff800`5442ead0
System Uptime: 0 days 0:00:00.010
nt!DebugService2+0x5:
fffff800`541ce0f5 cc              int     3
kd> g
KDTARGET: Refreshing KD connection
ExFAT: Invalid boot sector checksum.  Was 0x80cb6dde, should be 0xdbcb6dde.
ExFAT: Invalid boot sector checksum.  Was 0x80cb6dde, should be 0xdbcb6dde.
ExFAT: Invalid boot sector checksum.  Was 0x80cb6dde, should be 0xdbcb6dde.
ExFAT: Invalid boot sector checksum.  Was 0x80cb6dde, should be 0xdbcb6dde.
ExFAT: Invalid boot sector checksum.  Was 0x80cb6dde, should be 0xdbcb6dde.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
ExFAT: Invalid boot sector checksum.  Was 0x80ce4bde, should be 0x8dea9e84.
*BUSY* Debuggee is running...
```

Why Stop at Just One?



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

http://ddeville.me/2015/08/kernel-debugging-with-lldb-and-vmware-fusion

1. Install the macOS Kernel Debug Kit
   https://developer.apple.com/downloads
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?

https://opensource.apple.com/source/ntfs/ntfs-94/kext/ntfs_collate.c

```
/**
 * 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__);
        /*
```

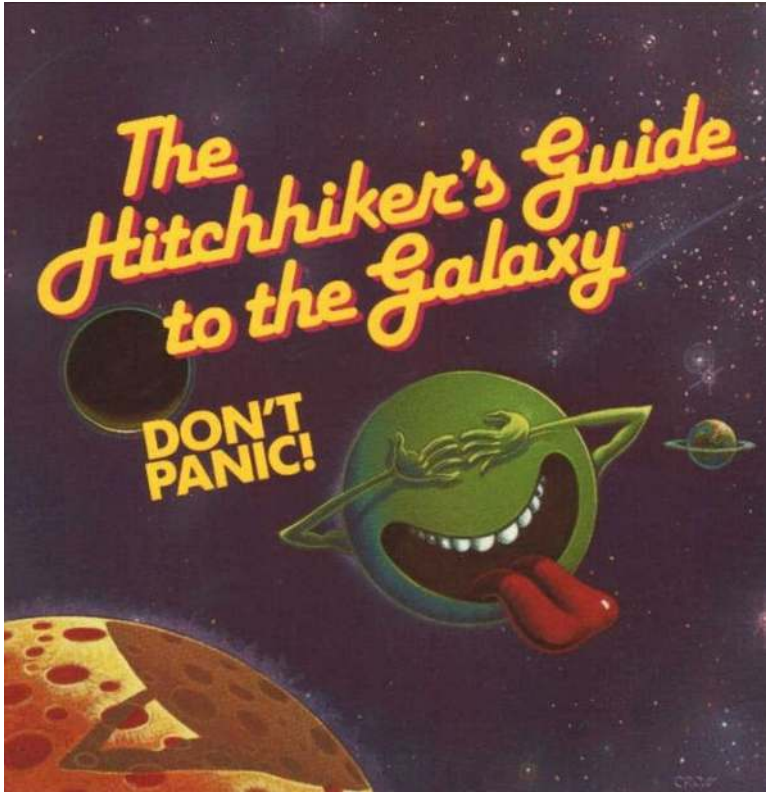
Why Does an OS Panic?

Something has gone wrong in the kernel, and we don't want memory corruption.

1. An access violation in the kernel
2. An explicit call to panic()

- macOS can never run from an NTFS drive
- Somebody plugged in a corrupt NTFS thumb drive

Perhaps Don’t Panic?

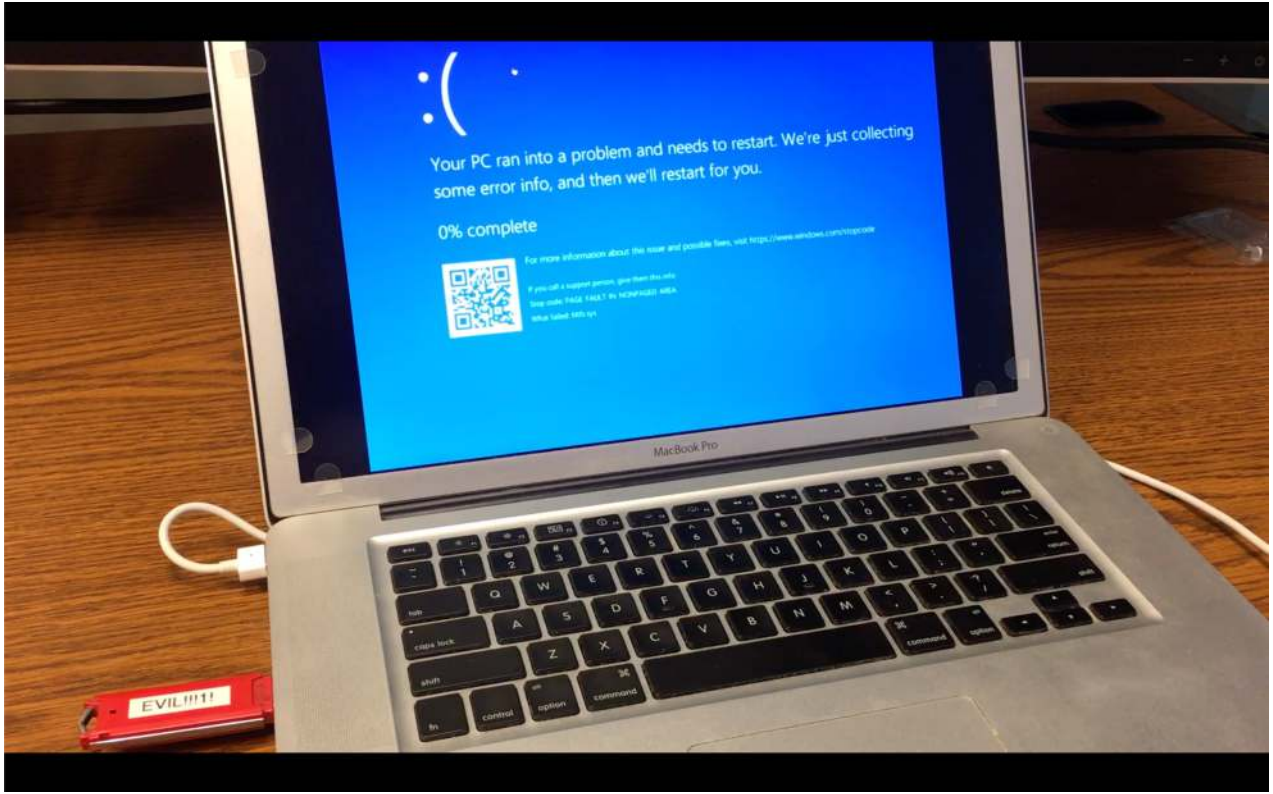


Death by Thumb Drive

Corrupt File System Attack Vectors

Do We Need Physical Access?

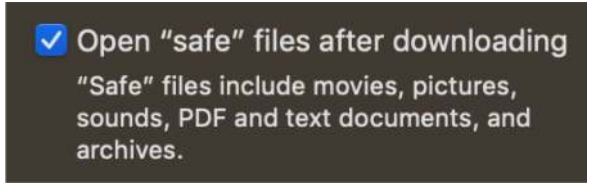
https://www.youtube.com/watch?v=r3MeifE2oFw



What About the Macs?

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

Safari auto-downloads DMG files

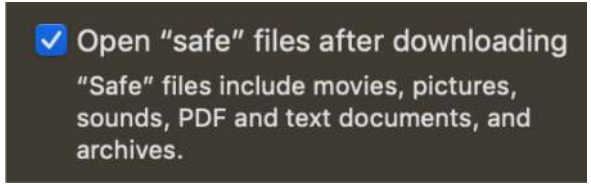


User double-clicks DMG

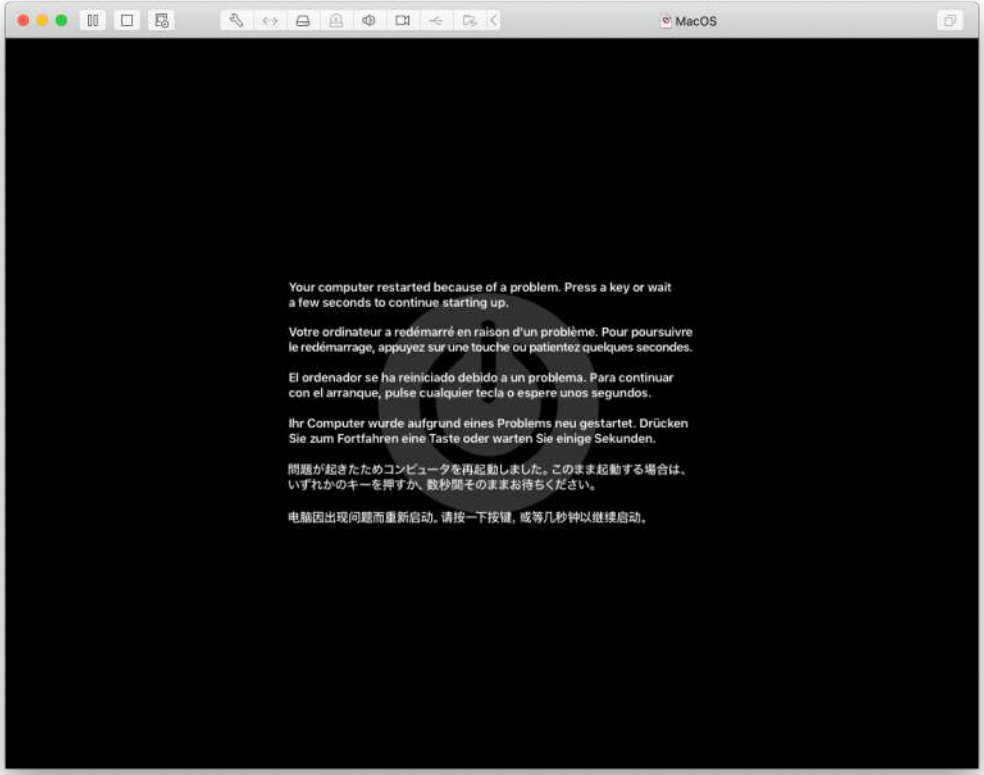
What About the Macs?

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

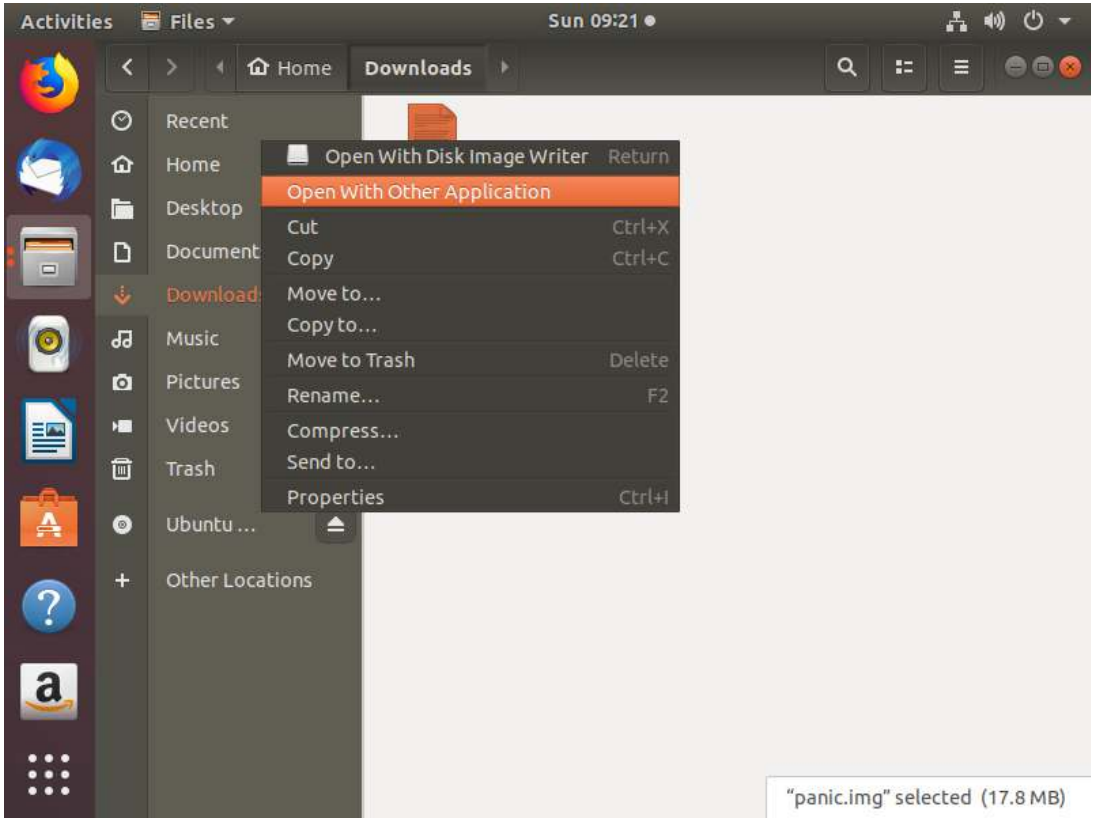
Safari auto-downloads DMG files



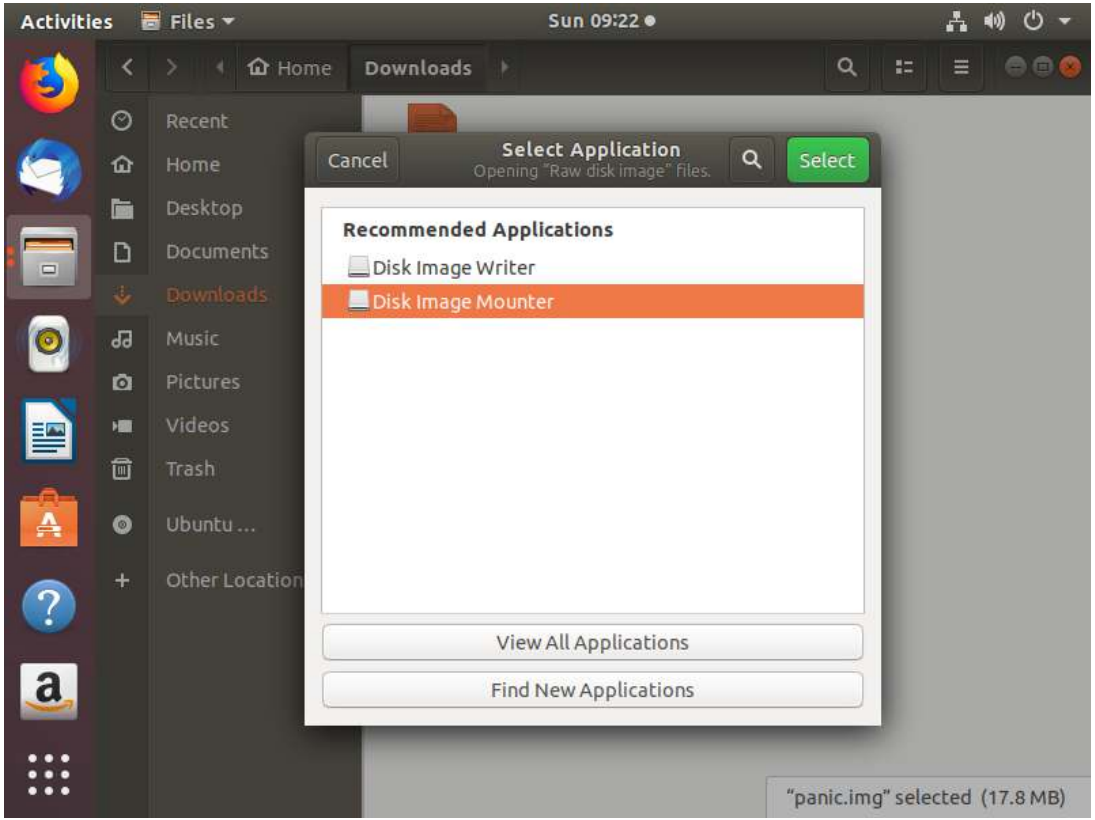
User double-clicks DMG



Ubuntu Linux?



Manual Interaction Required



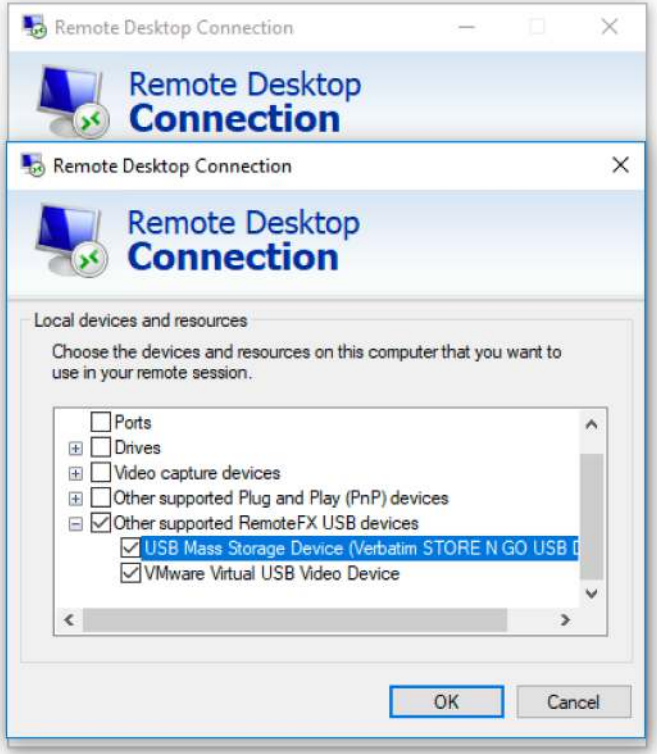
And Then…

```
[ 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
```

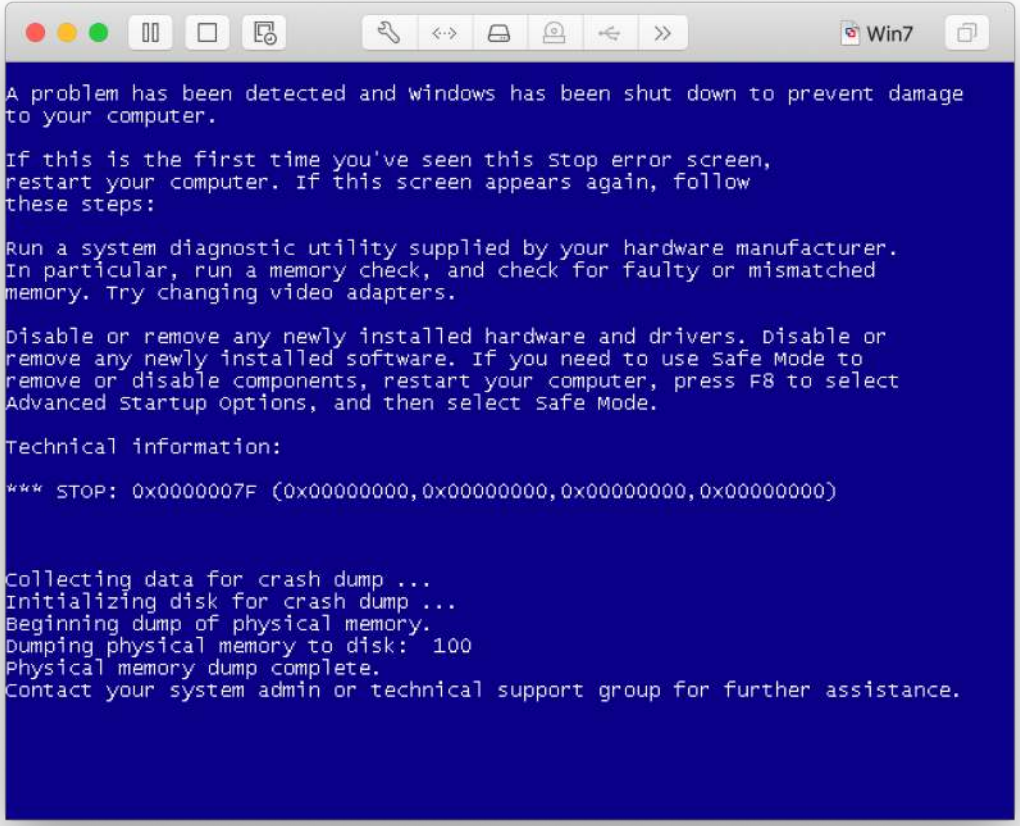
Windows RDP RemoteFX

RemoteFX allows USB Device pass-through

- Optional RDP feature
- Only for authenticated users



After Connecting USB via RemoteFX

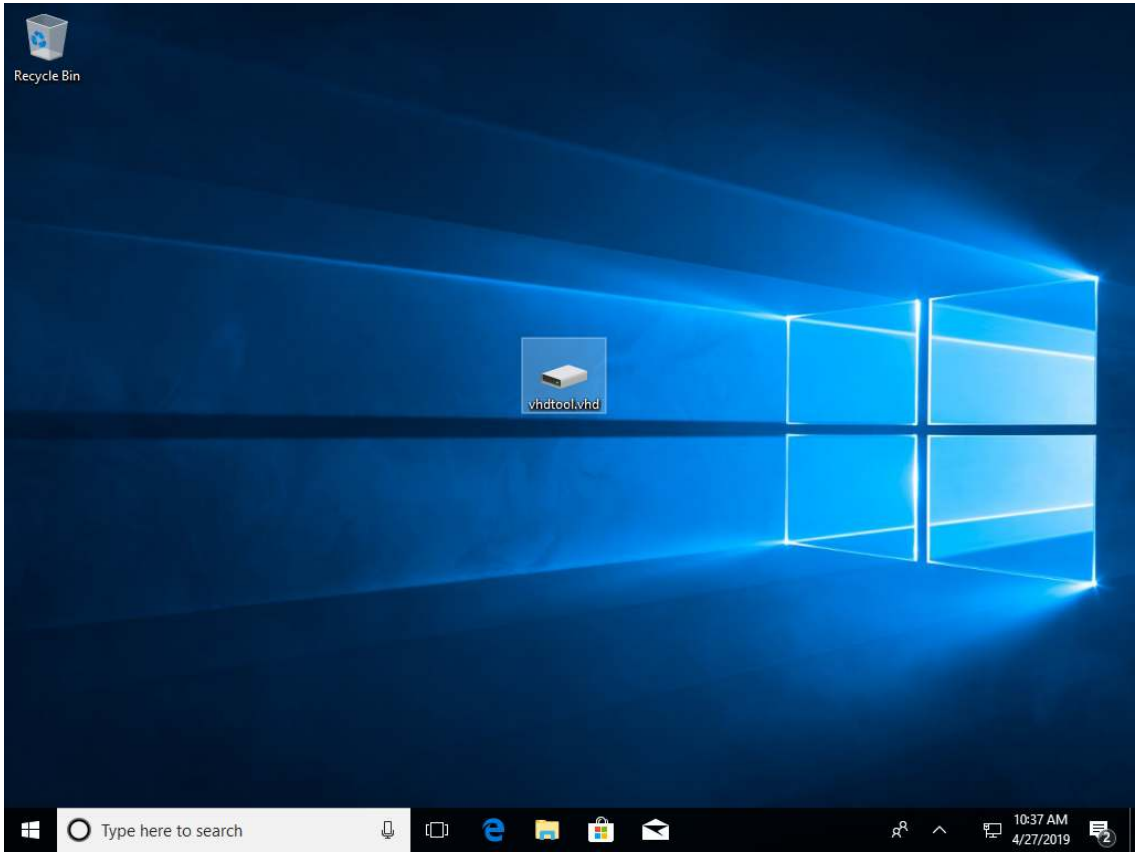


vhdtool

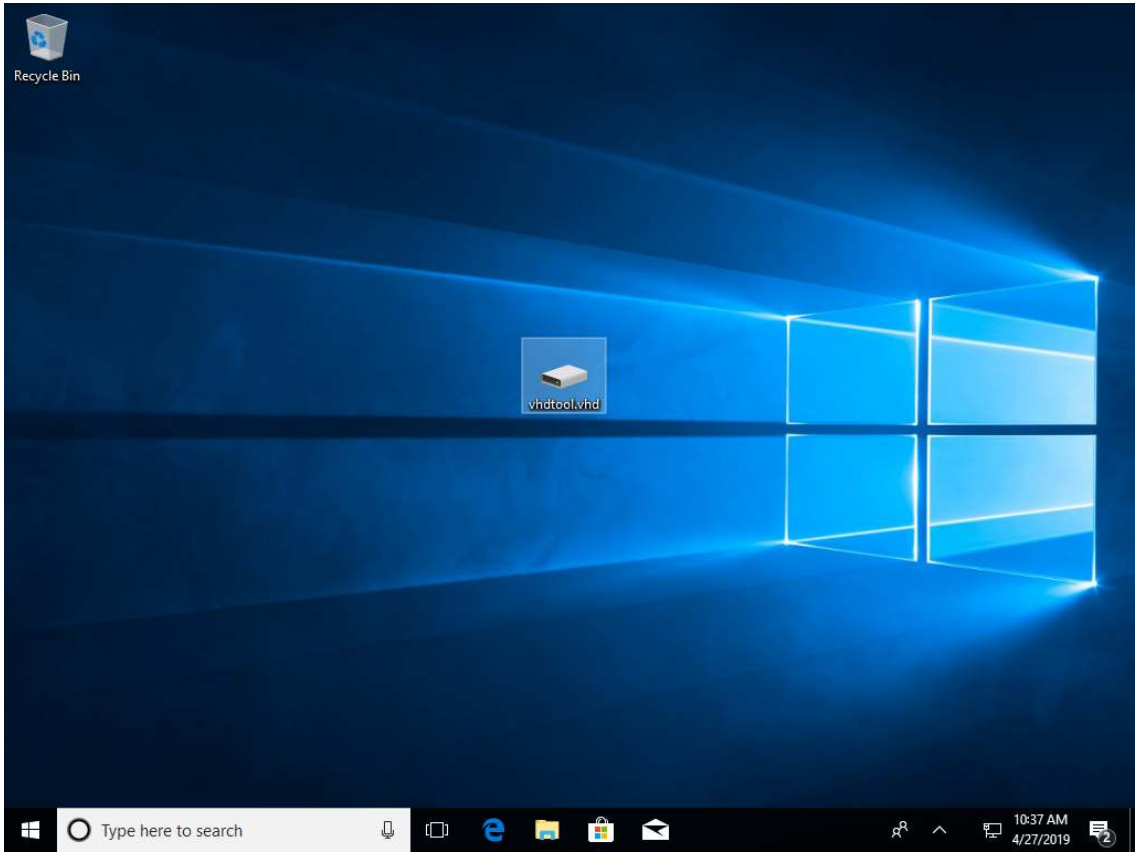
https://github.com/andreiw/vhdtool

```
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…



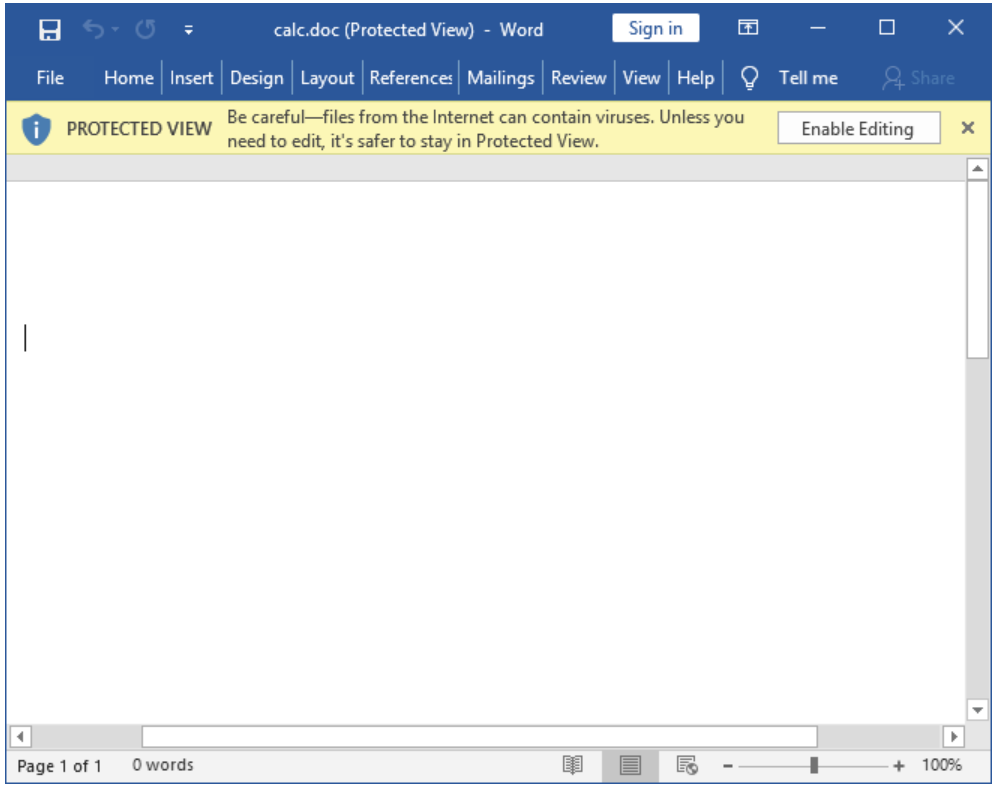
And Double-clicked it…

Are Your Security Products Scanning VHD(X)?

- **Yes**

- **No**

Mark of the Web



Mark of the Web

C:\WINDOWS\system32\cmd.exe

```
C:\Users\test_user\Downloads>dir /r
 Volume in drive C has no label.
 Volume Serial Number is B4C3-8A4B

 Directory of C:\Users\test_user\Downloads

06/11/2019  04:28 PM    <DIR>          .
06/11/2019  04:28 PM    <DIR>          ..
06/11/2019  04:27 PM        16,777,728 bsod.vhd
                               163 bsod.vhd Zone.Identifier:$DATA
               1 File(s)     16,777,728 bytes
               2 Dir(s)  41,660,596,224 bytes free

C:\Users\test_user\Downloads>_
```

Mark of the Web

C:\WINDOWS\system32\cmd.exe

bsod.vhd:Zone.Identifier - Notepad

File Edit Format View Help

```
[ZoneTransfer]
ZoneId=3
ReferrerUrl=http://attacker.example.com/index.php?dir=main/fsfuzz/bsods
HostUrl=http://attacker.example.com/main/fsfuzz/bsods/bsod.vhd
```

Windows (CRLF) | Ln 1, Col 1 | 100%

MotW and VHD(X)

- Mark of the Web is applied to downloaded VHD(X) files.
- Windows doesn’t treat downloaded VHD(X) files any differently.
- Security devices on the wire probably do not scan VHD(X) files.

Conclusion: VHD(X) files are a perfect vehicle for malicious payloads.

Death by Thumb Drive

Conclusion and Recommendations

Fuzzing OS Components with BFF

Normal BFF capabilities:

- Atomic iterations
- Crash de-duplication
- Exploitability determination
- Crash minimization and string minimization

Fuzzing anything OS-level:

- Cumulative effects
- Manual crash de-duplication
- Manual exploitability determination
- No crash minimization or string minimization

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/