```
#/*********************************************************************************************************
#* NEURONRAIN USB-md - Wireless Network and USB Stream Data Analytics
#* This program is free software: you can redistribute it and/or modify
#* it under the terms of the GNU General Public License as published by
#* the Free Software Foundation, either version 3 of the License, or
#* (at your option) any later version.
#*
#* This program is distributed in the hope that it will be useful,
#* but WITHOUT ANY WARRANTY; without even the implied warranty of
#* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#* GNU General Public License for more details.
#*
#* You should have received a copy of the GNU General Public License
#* along with this program.  If not, see <http://www.gnu.org/licenses/>.
#*
#--------------------------------------------------------------------------------------------------------
K.Srinivasan
#NeuronRain Documentation and Licensing: http://neuronrain-documentation.readthedocs.io/en/latest/
#Personal website(research): https://sites.google.com/site/kuja27/
#--------------------------------------------------------------------------------------------------------
#*********************************************************************************************************/
```

USBmd driver is an experimental modified version of already existing USB driver in linux.

Purpose of this modified version is for doing more sophisticated debugging of USB endpoints and devices and as USB packet sniffer. Technical Necessity for this was created due to prolonged data theft, id spoofing and cybercrime that has been happening in author's personal electronic devices for years that resulted in a Cybercrime Police Complaint also few years ago.

There were also such incidents while developing open source code (some code commits have description of these mysterious occurrences). There is no comprehensive USB debugger available on linux to sift bad traffic though there are strong evidences of such cybercrime and datatheft through other sources. Author is inclined to believe that such recurring events of datatheft that defies all logic can have no other intent but to cause malafide theft or loss of private data and an act of defamation among other things.

This is also done as a technical learning exercise to analyze USB Hosts, packets and USB's interaction,if any, with wireless devices including mobiles, wireless LANs(radiotap) etc.,

In the longterm USBmd might have to be integrated into VIRGO. As VIRGO would would have the synergy of AstroInfer machine learning codebase for "learning" from datasets, this USBmd driver can have the added ability of analyzing large USB traffic (as a dataset)

```
using some decision making algorithms and evolve as an anti-cybercrime, anti-plagiarism and anti-theft tool to single out
"malevolent" traffic that would save individuals and organisations from the travails of tampering and loss of sensitive
confidential data.

The pattern mining of numeric dataset designed for AstroInfer can apply here also since USB bitstream can be analyzed using
algorithms for
numerical dataset mining. Also Discrete Fourier Transform used for analyzing data for frequencies (periodicities if any)
can be used for
USB data , for example USB wireless traffic.

======================================================
new UMB driver bind - 27 Feb 2014 (for Bus id 7)
======================================================
Following example commandlines install umb.ko module, unbind the existing option driver from bus-device id and bind the
umb.ko to that bus id:

sudo insmod umb.ko
echo -n "7-1:1.0" > /sys/bus/usb/drivers/option/unbind
echo -n "7-1:1.0" > /sys/bus/usb/drivers/umb/bind

======================================================
1055. Commits as on 29 July 2014
======================================================
Driver has been ported and built on 3.15.5 kernel. Also a driver build script has been committed.

------------------------------------------------------
USBmd version 14.9.9 has been release tagged on 9 September 2014
------------------------------------------------------
------------------------------------------------------
USBmd version 15.1.8 has been release tagged on 8 January 2015
------------------------------------------------------

http://sourceforge.net/p/usb-md/code-
0/HEAD/tree/Adding%20new%20vendor%20and%20product%20IDs%20to%20an%20existing%20USB%20driver%20on%20Linux.html has steps to
add new vendor-id.

-------------------------------------------------------------------------------
1056. USB debug messages from "cat /sys/kernel/debug/usb/devices" for UMB bound above:
-------------------------------------------------------------------------------

T:  Bus=07 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 12 Spd=12   MxCh= 0
D:  Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=12d1 ProdID=140b Rev= 0.00
S:  Manufacturer=HUAÿWEI TECHNOLOGIES
```

```
S:  Product=HUAWEI Mobile
S:  SerialNumber=ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
C:* #Ifs= 4 Cfg#= 1 Atr=a0 MxPwr=500mA
I:* If#= 0 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff Driver=umb
E:  Ad=81(I) Atr=03(Int.) MxPS=  16 Ivl=128ms
E:  Ad=82(I) Atr=02(Bulk) MxPS=  64 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS=  64 Ivl=0ms
I:* If#= 1 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=option
E:  Ad=84(I) Atr=02(Bulk) MxPS=  64 Ivl=0ms
E:  Ad=04(O) Atr=02(Bulk) MxPS=  64 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=option
E:  Ad=86(I) Atr=02(Bulk) MxPS=  64 Ivl=0ms
E:  Ad=06(O) Atr=02(Bulk) MxPS=  64 Ivl=0ms
I:* If#= 3 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50 Driver=usb-storage
E:  Ad=87(I) Atr=02(Bulk) MxPS=  64 Ivl=0ms
E:  Ad=08(O) Atr=02(Bulk) MxPS=  64 Ivl=0ms

---------------------------------------------------------------------
1057. usbmon, libpcap tcpdump and wireshark (or vusb-analyzer) debugging
---------------------------------------------------------------------
*mount -t debugfs none_debugs /sys/kernel/debug
*modprobe usbmon
*ls /sys/kernel/debug/usb/usbmon/

0s  0u  1s  1t  1u  2s  2t  2u  3s  3t  3u  4s  4t  4u  5s  5t  5u  6s  6t  6u  7s  7t  7u  8s  8t  8u

*cat /sys/kernel/debug/usb/usbmon/8t > usbmon.mon (any of the above usbmon debug logs)
*vusb-analyzer usbmon.mon

ef728540 3811287714 S Ci:001:00 s a3 00 0000 0001 0004 4 <
ef728540 3811287743 C Ci:001:00 0 4 = 00010000
ef728540 3811287752 S Ci:001:00 s a3 00 0000 0002 0004 4 <
ef728540 3811287763 C Ci:001:00 0 4 = 00010000
f50f6540 3811287770 S Ii:001:01 -115 2 <
f50f6540 3811287853 C Ii:001:01 -2 0
f5390540 3814543695 S Ci:001:00 s a3 00 0000 0001 0004 4 <
f5390540 3814543715 C Ci:001:00 0 4 = 00010000
f5390540 3814543756 S Ci:001:00 s a3 00 0000 0002 0004 4 <
f5390540 3814543767 C Ci:001:00 0 4 = 00010000
f50f6540 3814543805 S Ii:001:01 -115 2 <

*modprobe usbmon
*ls /dev/usbmon[1-8]
*tcpdump -i usbmon1 -w usbmon.pcap
```

```
tcpdump: listening on usbmon1, link-type USB_LINUX_MMAPPED (USB with padded Linux header), capture size 65535 bytes
^C86 packets captured
86 packets received by filter

*wireshark usbmon.pcap (loads on wireshark)

----------------------------------------------------------
1058. Dynamic Debug - dev_dbg() and dev_vdbg()
----------------------------------------------------------

USB Debugging References:
-------------------------
- Texas Instruments - http://elinux.org/images/1/17/USB_Debugging_and_Profiling_Techniques.pdf

---------------------------------------------------------
NeuronRain version 15.6.15 release tagged
---------------------------------------------------------


---------------------------------------------------------
1059. Commits as on 11 July 2015
---------------------------------------------------------
usbmd kernel module has been ported to Linux Kernel 4.0.5

---------------------------------------------------------
1060. Commits as on 26 November 2015
---------------------------------------------------------
- Updated USB-md driver with a lookup of VIRGO kernel_analytics config variable exported by kernel_analytics module in
umb_read() as default.
- New header file umb.h has been added that externs the VIRGO kernel_analytics config array variables
- Module.symvers has been imported from VIRGO kernel_analytics and clean target has been commented in build script after
initial build as make clean removes Module.symvers.
- kern.log with umb_read() and umb_write() have been added with following commandlines:
      - cat /dev/umb0 - invokes umb_read() but there are kernel panics sometimes
      - cat <file> > /dev/umb0 - invokes umb_write()
  where umb0 is usb-md device name registered with /sys/bus/usb as below:
      - insmod umb.ko
      - echo -n "7-1:1.0" > /sys/bus/usb/drivers/option/unbind
      - echo -n "7-1:1.0" > /sys/bus/usb/drivers/umb/bind
- Updated build generated sources and object files have been added

----------------------------------------------------------
1061. Commits as on 27 November 2015
----------------------------------------------------------
New folder usb_wwan_modified has been added that contains the USB serial, option and wireless USB modem WWAN drivers from
```

kernel mainline
instrumented with lot of printk()s so that log messages are written to kern.log. Though dev_dbg dynamic debugging can be used by writing to /sys/kernel/debug/<...>/dynamic_debug
printk()s are sufficient for now. This traces through the USB connect and data transfer code:

- probe
- buffer is copied from userspace to kernelspace
- URB is allocated in kernel
- buffer is memcopied to URB
- usb send/receive bulk pipe calls
- usb_fill_bulk_urb

Almost all buffers like in and out buffers in URBs, portdata, interfacedata, serial_data, serial_port_data are printed to kern.log. This log is
analyzable by AsFer machine learning code for USB debugging similar to usbmon logs.

These are initial commits only and usb-serial.c, usb_wwan.c, option.c and serial.h might be significantly altered going forward.

1062. Commits as on 30 November 2015

Added usb.h from kernel mainline, instrumented with printk() to print transfer_buffer in
usb_fill_[control/bulk/interrupt]_urb() functions. kern.log for this has been added in usb_wwan_modified/testlogs.

1063. Commits as on 1 December 2015

- new kernel function print_buffer() has been added in usb.h that prints contents of char buffer in hex
- Above print_buffer() is invoked to print transfer_buffer in usb_wwan.c, usb-serial.c, option.c
- kern.log with print_buffer() output has been added - This dumps similar to wireshark, usbmon and other usb analyzers.

1064. Commits as on 2 December 2015

- changed print_buffer() printk() to print a delimiter in each byte for AsFer Machine Learning code processing
- add a parser script for kern.log to print print_buffer() lines
- parsed kern.log with print_buffer() lines has been added
- Added an Apache Spark MapReduce python script to compute byte frequency in parsed print_buffer() kern.log

879. (FEATURE) NeuronRain USBmd Debug and Malafide Traffic Analytics

As mentioned in commit notes above, USB incoming and outgoing data transfer_buffer are dumped byte-by-byte. Given this data various
analytics can be performed most of which are already implemented in AsFer codebase:

- frequency of bytes
- most frequent sequence of bytes
- bayesian and decision tree inference
- deep learning
- perceptrons
- streaming algorithms for USB data stream

and so on.

1065. Commits as on 3 December 2015

- Apache Spark script for analyzing the USBWWAN byte stream logs has been updated with byte counts map-reduce functions from print_buffer() logs
and temp DataFrame Table creation with SparkSQL.
- logs for the script have been added in usb_wwan_modified/python-src/testlogs/Spark_USBWWANLogMapReduceParser.out.3December2015
- kern.log parser shellscript has been updated

1066. AsFer commits for USBmd as on 4 December 2015

All the Streaming_<>.py Streaming Algorithm implementations in AsFer/python-src/ have been updated with:

- hashlib ripemd160 hash MD algorithm for hash functions and return hexdigest()
- USBWWAN byte stream data from USBmd print_buffer() logs in usb-md/usb_wwan_modified/testlogs/ has been added as a Data Storage and Data Source
- logs for the above have been added to asfer/python-src/testlogs/
- Streaming Abstract Generator has been updated with USB stream data iterable and parametrized for data source and storage
- Some corrections to the asfer/python-src/Streaming_<> scripts

1067. Commits as on 7 December 2015

- added Spark Mapreduce and DataFrame log for USBWWAN byte stream
- added a parsed kern.log with only bytes from USBWWAN stream
- Added dict() and sort() for query results and printed cardinality of the stream data set which is the size of the dict.
An example log has been added which prints the cardinality as ~250. In contrast, LogLog and HyperLogLog counter estimations approximate the cardinality to 140 and 110 respectively

880. (FEATURE) AsFer commits for USBmd as on 11 December 2015 - USBWWAN stream data backend in MongoDB

Dependency Injection code commits for MongoDB backend - With this MongoDB is also a storage backend for AsFer algorithms similar to MySQL:

- Abstract_DBBackend.py has been updated for both MySQL and MongoDB injections

- MongoDB configuration and backend connect/query code has been added. Backend is either populated by Robomongo or pymongo reading from the Streaming Abstract Generator iterable framework.
- With this AsFer supports both SQL(MySQL) and NoSQL(file,hive,hbase,cassandra backends in Streaming Abstract Generator).
- log with a simple NoSQL table with StreamingData.txt and USBWWAN data has been added to testlogs/.
- MongoDB configuration has a database(asfer-database) and a collection(asfer-collection).
- MongoDB_DBBackend @provides pymongo.collection.Collection which is @inject-ed to Abstract_DBBackend

----

1068. Commits as on 10 January 2016

NeuronRain USBmd enterprise version 2016.1.10 released.

1069. Commits - 4 August 2016

1.New build script for drivers/usb top level folder has been added.
2.Copyleft notices updated
3.print_buffer() in usb.h has been #ifdef-ed based on a build time flag to suppress the buffer bytes dump preferentially so that kern.log is not flooded.
4.Flag PRINT_BUFFER has to be defined with #define somewhere within KBuild makefiles or externally.
5..ko files rebuilt
6. Miscellaneous code changes to suppress kbuild warnings - cast etc.,
7. PRINT_BUFFER block changed to print the bytes in single line for each buffer

----

1070. Commits - 13 July 2017 - usb-storage driver last sector access slab out of bounds error in 64-bit - committed for analysis
- this error was frequently witnessed in VIRGO 32-bit stability issues and panics - ISRA looks like a GCC optimization of a function invocation (Interprocedural Scalar Replacement of Aggregates)

```
Jul 13 15:03:36 localhost kernel: [ 9837.497280] ==================================================================
Jul 13 15:03:36 localhost kernel: [ 9837.499787] ==================================================================
Jul 13 15:03:36 localhost kernel: [ 9837.499822] BUG: KASAN: slab-out-of-bounds in
last_sector_hacks.isra.1.part.2+0xc9/0x1d0 [usb_storage] at addr ffff88007cdaa758
Jul 13 15:03:36 localhost kernel: [ 9837.499831] Read of size 8 by task usb-storage/6243
Jul 13 15:03:36 localhost kernel: [ 9837.499844] CPU: 0 PID: 6243 Comm: usb-storage Tainted: G    B           4.10.3 #18
Jul 13 15:03:36 localhost kernel: [ 9837.499849] Hardware name: Dell Inc. Inspiron 1545                   /0J037P, BIOS A14
12/07/2009
Jul 13 15:03:36 localhost kernel: [ 9837.499851] Call Trace:
Jul 13 15:03:36 localhost kernel: [ 9837.499863]  dump_stack+0x63/0x8b
Jul 13 15:03:36 localhost kernel: [ 9837.499870]  kasan_object_err+0x21/0x70
Jul 13 15:03:36 localhost kernel: [ 9837.499877]  kasan_report.part.1+0x219/0x4f0
```

```
 Jul 13 15:03:36 localhost kernel: [ 9837.499893]  ? last_sector_hacks.isra.1.part.2+0xc9/0x1d0 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.499899]  kasan_report+0x25/0x30
 Jul 13 15:03:36 localhost kernel: [ 9837.499906]  __asan_load8+0x5e/0x70
 Jul 13 15:03:36 localhost kernel: [ 9837.499922]  last_sector_hacks.isra.1.part.2+0xc9/0x1d0 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.499938]  usb_stor_invoke_transport+0x1a1/0x960 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.499946]  ? migrate_swap_stop+0x2e0/0x2e0
 Jul 13 15:03:36 localhost kernel: [ 9837.499963]  ? usb_stor_port_reset+0xb0/0xb0 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.499973]  ? wait_for_completion_interruptible+0x1a7/0x260
 Jul 13 15:03:36 localhost kernel: [ 9837.499981]  ? wait_for_completion_killable+0x2a0/0x2a0
 Jul 13 15:03:36 localhost kernel: [ 9837.499989]  ? raise_softirq_irqoff+0xba/0xd0
 Jul 13 15:03:36 localhost kernel: [ 9837.499995]  ? wake_up_q+0x80/0x80
 Jul 13 15:03:36 localhost kernel: [ 9837.500011]  usb_stor_transparent_scsi_command+0xe/0x10 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  usb_stor_control_thread+0x344/0x510 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ? usb_stor_disconnect+0x120/0x120 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ? default_wake_function+0x2f/0x40
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ? __wake_up_common+0x78/0xc0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kthread+0x178/0x1d0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ? usb_stor_disconnect+0x120/0x120 [usb_storage]
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ? kthread_create_on_node+0xd0/0xd0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ret_from_fork+0x2c/0x40
 Jul 13 15:03:36 localhost kernel: [ 9837.500017] Object at ffff88007cdaa668, in cache kmalloc-192 size: 192
 Jul 13 15:03:36 localhost kernel: [ 9837.500017] Allocated:
 Jul 13 15:03:36 localhost kernel: [ 9837.500017] PID = 6277
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  save_stack_trace+0x1b/0x20
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  save_stack+0x46/0xd0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kasan_kmalloc+0xad/0xe0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kmem_cache_alloc_trace+0xef/0x210
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kernfs_fop_open+0x14b/0x540
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  do_dentry_open+0x39a/0x560
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  vfs_open+0x84/0xd0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  path_openat+0x4ab/0x1e10
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  do_filp_open+0x122/0x1c0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  do_sys_open+0x17c/0x2c0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  compat_SyS_open+0x1b/0x20
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  do_fast_syscall_32+0x188/0x300
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  entry_SYSENTER_compat+0x4c/0x5b
 Jul 13 15:03:36 localhost kernel: [ 9837.500017] Freed:
 Jul 13 15:03:36 localhost kernel: [ 9837.500017] PID = 6277
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  save_stack_trace+0x1b/0x20
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  save_stack+0x46/0xd0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kasan_slab_free+0x71/0xb0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kfree+0x9e/0x1e0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  kernfs_fop_release+0x87/0xa0
 Jul 13 15:03:36 localhost kernel: [ 9837.500017]  __fput+0x177/0x350
```

```
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ____fput+0xe/0x10
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  task_work_run+0xa0/0xc0
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  exit_to_usermode_loop+0xc5/0xd0
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  do_fast_syscall_32+0x2ef/0x300
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  entry_SYSENTER_compat+0x4c/0x5b
Jul 13 15:03:36 localhost kernel: [ 9837.500017] Memory state around the buggy address:
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ffff88007cdaa600: fc fc fc fc fc fc fc fc fc fc fc fc fc fb fb fb
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ffff88007cdaa680: fb fb fb fb fb fb fb fb fb fb fb fb fb fb fb fb
Jul 13 15:03:36 localhost kernel: [ 9837.500017] >ffff88007cdaa700: fb fb fb fb fb fc fc fc fc fc fc fc fc fc fc fc
Jul 13 15:03:36 localhost kernel: [ 9837.500017]                                                      ^
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ffff88007cdaa780: fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc
Jul 13 15:03:36 localhost kernel: [ 9837.500017]  ffff88007cdaa800: fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc
Jul 13 15:03:36 localhost kernel: [ 9837.500017] ==================================================================
Jul 13 15:03:37 localhost kernel: [ 9837.668157] ==================================================================
Jul 13 15:03:37 localhost kernel: [ 9837.668191] BUG: KASAN: slab-out-of-bounds in
last_sector_hacks.isra.1.part.2+0xc9/0x1d0 [usb_storage] at addr ffff88007cdaa758
Jul 13 15:03:37 localhost kernel: [ 9837.668200] Read of size 8 by task usb-storage/6243
Jul 13 15:03:37 localhost kernel: [ 9837.668213] CPU: 1 PID: 6243 Comm: usb-storage Tainted: G    B           4.10.3 #18
Jul 13 15:03:37 localhost kernel: [ 9837.668218] Hardware name: Dell Inc. Inspiron 1545                   /0J037P, BIOS A14
12/07/2009
Jul 13 15:03:37 localhost kernel: [ 9837.668220] Call Trace:
Jul 13 15:03:37 localhost kernel: [ 9837.668233]  dump_stack+0x63/0x8b
Jul 13 15:03:37 localhost kernel: [ 9837.668240]  kasan_object_err+0x21/0x70
Jul 13 15:03:37 localhost kernel: [ 9837.668247]  kasan_report.part.1+0x219/0x4f0
Jul 13 15:03:37 localhost kernel: [ 9837.668263]  ? last_sector_hacks.isra.1.part.2+0xc9/0x1d0 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668269]  kasan_report+0x25/0x30
Jul 13 15:03:37 localhost kernel: [ 9837.668277]  __asan_load8+0x5e/0x70
Jul 13 15:03:37 localhost kernel: [ 9837.668292]  last_sector_hacks.isra.1.part.2+0xc9/0x1d0 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668308]  usb_stor_invoke_transport+0x1a1/0x960 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668316]  ? migrate_swap_stop+0x2e0/0x2e0
Jul 13 15:03:37 localhost kernel: [ 9837.668332]  ? usb_stor_port_reset+0xb0/0xb0 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668343]  ? wait_for_completion_interruptible+0x1a7/0x260
Jul 13 15:03:37 localhost kernel: [ 9837.668351]  ? wait_for_completion_killable+0x2a0/0x2a0
Jul 13 15:03:37 localhost kernel: [ 9837.668360]  ? raise_softirq_irqoff+0xba/0xd0
Jul 13 15:03:37 localhost kernel: [ 9837.668366]  ? wake_up_q+0x80/0x80
Jul 13 15:03:37 localhost kernel: [ 9837.668382]  usb_stor_transparent_scsi_command+0xe/0x10 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668398]  usb_stor_control_thread+0x344/0x510 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668415]  ? usb_stor_disconnect+0x120/0x120 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668422]  ? default_wake_function+0x2f/0x40
Jul 13 15:03:37 localhost kernel: [ 9837.668430]  ? __wake_up_common+0x78/0xc0
Jul 13 15:03:37 localhost kernel: [ 9837.668436]  kthread+0x178/0x1d0
Jul 13 15:03:37 localhost kernel: [ 9837.668454]  ? usb_stor_disconnect+0x120/0x120 [usb_storage]
Jul 13 15:03:37 localhost kernel: [ 9837.668460]  ? kthread_create_on_node+0xd0/0xd0
Jul 13 15:03:37 localhost kernel: [ 9837.668466]  ret_from_fork+0x2c/0x40
Jul 13 15:03:37 localhost kernel: [ 9837.668472] Object at ffff88007cdaa668, in cache kmalloc-192 size: 192
Jul 13 15:03:37 localhost kernel: [ 9837.668478] Allocated:
Jul 13 15:03:37 localhost kernel: [ 9837.668483] PID = 6277
Jul 13 15:03:37 localhost kernel: [ 9837.668494]  save_stack_trace+0x1b/0x20
Jul 13 15:03:37 localhost kernel: [ 9837.668500]  save_stack+0x46/0xd0
Jul 13 15:03:37 localhost kernel: [ 9837.668506]  kasan_kmalloc+0xad/0xe0
Jul 13 15:03:37 localhost kernel: [ 9837.668513]  kmem_cache_alloc_trace+0xef/0x210
Jul 13 15:03:37 localhost kernel: [ 9837.668520]  kernfs_fop_open+0x14b/0x540
Jul 13 15:03:37 localhost kernel: [ 9837.668527]  do_dentry_open+0x39a/0x560
Jul 13 15:03:37 localhost kernel: [ 9837.668532]  vfs_open+0x84/0xd0
Jul 13 15:03:37 localhost kernel: [ 9837.668538]  path_openat+0x4ab/0x1e10
Jul 13 15:03:37 localhost kernel: [ 9837.668544]  do_filp_open+0x122/0x1c0
Jul 13 15:03:37 localhost kernel: [ 9837.668549]  do_sys_open+0x17c/0x2c0
Jul 13 15:03:37 localhost kernel: [ 9837.668554]  compat_SyS_open+0x1b/0x20
Jul 13 15:03:37 localhost kernel: [ 9837.668561]  do_fast_syscall_32+0x188/0x300
Jul 13 15:03:37 localhost kernel: [ 9837.668568]  entry_SYSENTER_compat+0x4c/0x5b
Jul 13 15:03:37 localhost kernel: [ 9837.668570] Freed:
Jul 13 15:03:37 localhost kernel: [ 9837.668575] PID = 6277
Jul 13 15:03:37 localhost kernel: [ 9837.668583]  save_stack_trace+0x1b/0x20
Jul 13 15:03:37 localhost kernel: [ 9837.668589]  save_stack+0x46/0xd0
Jul 13 15:03:37 localhost kernel: [ 9837.668594]  kasan_slab_free+0x71/0xb0
Jul 13 15:03:37 localhost kernel: [ 9837.668599]  kfree+0x9e/0x1e0
Jul 13 15:03:37 localhost kernel: [ 9837.668605]  kernfs_fop_release+0x87/0xa0
Jul 13 15:03:37 localhost kernel: [ 9837.668611]  __fput+0x177/0x350
Jul 13 15:03:37 localhost kernel: [ 9837.668616]  ____fput+0xe/0x10
Jul 13 15:03:37 localhost kernel: [ 9837.668623]  task_work_run+0xa0/0xc0
Jul 13 15:03:37 localhost kernel: [ 9837.668629]  exit_to_usermode_loop+0xc5/0xd0
Jul 13 15:03:37 localhost kernel: [ 9837.668635]  do_fast_syscall_32+0x2ef/0x300
Jul 13 15:03:37 localhost kernel: [ 9837.668642]  entry_SYSENTER_compat+0x4c/0x5b
Jul 13 15:03:37 localhost kernel: [ 9837.668644] Memory state around the buggy address:
Jul 13 15:03:37 localhost kernel: [ 9837.668655]  ffff88007cdaa600: fc fc fc fc fc fc fc fc fc fc fc fc fc fb fb fb
Jul 13 15:03:37 localhost kernel: [ 9837.668664]  ffff88007cdaa680: fb fb fb fb fb fb fb fb fb fb fb fb fb fb fb fb
Jul 13 15:03:37 localhost kernel: [ 9837.668674] >ffff88007cdaa700: fb fb fb fb fb fc fc fc fc fc fc fc fc fc fc fc
Jul 13 15:03:37 localhost kernel: [ 9837.668680]                                                      ^
Jul 13 15:03:37 localhost kernel: [ 9837.668689]  ffff88007cdaa780: fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc
Jul 13 15:03:37 localhost kernel: [ 9837.668698]  ffff88007cdaa800: fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc
Jul 13 15:03:37 localhost kernel: [ 9837.668704] ==================================================================
Jul 13 15:03:37 localhost NetworkManager[745]: <info>  [1499938417.1889]   address 192.168.1.100

---------------------------------------------------------------------------------------------------------------------------
--------
```

1071. Commits - 13 August 2017 - Suspicious use-after-free error flagged by Kernel Address Sanitizer - committed for analysis

This error precedes last_sector_hacks ISRA error above in USB storage driver.

```
---------------------------------------------------------------------------------------------------------------------------
--------
Aug 13 14:53:17 localhost kernel: [   47.797146] BUG: KASAN: use-after-free in sr_probe+0x7e0/0xb20 at addr
ffff88000009637e
Aug 13 14:53:17 localhost kernel: [   47.797146] Read of size 1 by task kworker/u4:1/37
Aug 13 14:53:17 localhost kernel: [   47.797146] page:ffffea0000002580 count:0 mapcount:0 mapping:          (null)
index:0x0
Aug 13 14:53:17 localhost kernel: [   47.797146] flags: 0x0()
Aug 13 14:53:17 localhost kernel: [   47.797146] raw: 0000000000000000 0000000000000000 0000000000000000 00000000ffffffff
Aug 13 14:53:17 localhost kernel: [   47.797146] raw: ffffea00000025a0 ffffea00000025a0 0000000000000000 0000000000000000
Aug 13 14:53:17 localhost kernel: [   47.797146] page dumped because: kasan: bad access detected
Aug 13 14:53:17 localhost kernel: [   47.797146] CPU: 1 PID: 37 Comm: kworker/u4:1 Tainted: G    B           4.10.3 #18
Aug 13 14:53:17 localhost kernel: [   47.797146] Hardware name: Dell Inc. Inspiron 1545                   /0J037P, BIOS A14
12/07/2009
Aug 13 14:53:17 localhost kernel: [   47.797146] Workqueue: events_unbound async_run_entry_fn
Aug 13 14:53:17 localhost kernel: [   47.797146] Call Trace:
Aug 13 14:53:17 localhost kernel: [   47.797146]  dump_stack+0x63/0x8b
Aug 13 14:53:17 localhost kernel: [   47.797146]  kasan_report.part.1+0x4bc/0x4f0
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? sr_probe+0x7e0/0xb20
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? scsi_mode_select+0x370/0x370
Aug 13 14:53:17 localhost kernel: [   47.797146]  kasan_report+0x25/0x30
Aug 13 14:53:17 localhost kernel: [   47.797146]  __asan_load1+0x47/0x50
Aug 13 14:53:17 localhost kernel: [   47.797146]  sr_probe+0x7e0/0xb20
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? kernfs_next_descendant_post+0x93/0xf0
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? sr_block_ioctl+0xe0/0xe0
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? sysfs_do_create_link_sd.isra.2+0x7c/0xc0
Aug 13 14:53:17 localhost kernel: [   47.797146]  driver_probe_device+0x40b/0x670
Aug 13 14:53:17 localhost kernel: [   47.797146]  __device_attach_driver+0xd9/0x160
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? __driver_attach+0x120/0x120
Aug 13 14:53:17 localhost kernel: [   47.797146]  bus_for_each_drv+0x107/0x180
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? bus_rescan_devices+0x20/0x20
Aug 13 14:53:17 localhost kernel: [   47.797146]  __device_attach+0x17e/0x200
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? device_bind_driver+0x80/0x80
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? kobject_uevent_env+0x1ec/0x7f0
Aug 13 14:53:17 localhost kernel: [   47.797146]  device_initial_probe+0x13/0x20
Aug 13 14:53:17 localhost kernel: [   47.797146]  bus_probe_device+0xfe/0x120
Aug 13 14:53:17 localhost kernel: [   47.797146]  device_add+0x5f1/0x9f0
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? device_private_init+0xc0/0xc0
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? scsi_dh_add_device+0xd4/0x130
Aug 13 14:53:17 localhost kernel: [   47.797146]  scsi_sysfs_add_sdev+0xd1/0x350
Aug 13 14:53:17 localhost kernel: [   47.797146]  do_scan_async+0xfd/0x230
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? scsi_scan_host+0x250/0x250
Aug 13 14:53:17 localhost kernel: [   47.797146]  async_run_entry_fn+0x84/0x270
Aug 13 14:53:17 localhost kernel: [   47.797146]  ? pwq_dec_nr_in_flight+0x8c/0x110
Aug 13 14:53:17 localhost kernel: [   47.797146]  process_one_work+0x2c6/0x7d0
Aug 13 14:53:17 localhost kernel: [   47.797146]  worker_thread+0x90/0x850
Aug 13 14:53:17 localhost kernel: [   47.797146]  kthread+0x178/0x1d0

---------------------------------------------------------------------------------------------------------------------------
--------------
881. (FEATURE-DONE) Spark Cloud Analytics for Linux Kernel 4.10.3 64 bit with Kernel Address Sanitizer debug logging
enabled - 15 August 2017
- Commits 1
---------------------------------------------------------------------------------------------------------------------------
--------------
(*) Upgraded Spark version to 2.1.0 on Hadoop 2.7
(*) Changed to SparkContext text file instead of reading the input kernel log in python I/O
(*) Added flatMap to front of MapReduce chain of transformations for tokenizer
(*) Changed the input kernel log to 64bit 4.10.3 Kernel Address Sanitizer enabled kern.log which prints lot of debugging
information on
memory accesses especially for USBWWAN and USB Storage drivers.
(*) This is an alternative to traditional promiscuous USB Analyzers like WireShark to get kernel stack traces for USB and
WLAN operations.
(*) Particularly useful in malware related untoward memory access and traffic analysis
(*) Unifies Kernel Address Sanitizer, USB storage/WLAN driver and Spark Cloud for analytics
(*) Logs for this have been committed to testlogs/ and python-src/testlogs

---------------------------------------------------------------------------------------------------------------------------
--------------
882. (FEATURE-DONE) Spark Cloud Analytics for Linux Kernel 4.10.3 64 bit with Kernel Address Sanitizer debug logging
enabled - 15 August 2017
- Commits 2
---------------------------------------------------------------------------------------------------------------------------
--------------
(*) Added a substring match filter to RDD map/reduce transformations chain
(*) Presently hardcoded as "+0x" which extracts all kernel functions invoked from Kernel Address Sanitizer kern.log and
their frequencies

Previous profiling prints following top kernel function invocations:
(u'last_sector_hacks.isra.1.part.2+0xc9/0x1d0', 159),
 (u'usb_stor_disconnect+0x120/0x120', 106),
 (u'save_stack+0x46/0xd0', 106),
 (u'save_stack_trace+0x1b/0x20', 106),
 (u'entry_SYSENTER_compat+0x4c/0x5b', 85),
 (u'kthread+0x178/0x1d0', 74),
implying heavy dependence on last_sector_hacks.isra gcc optimization. Discussion on
https://groups.google.com/forum/#!topic/linux.kernel/IYBXrW7K2Vc shows it to be an old kernel bug.
```

883. (FEATURE-DONE) Commits - 24 September 2017 - USB-md driver for USB and Wireless LAN analytics for 4.13.3 64-bit kernel

(*) USB-md driver in GitHub and SourceForge at present are 32-bit based on mainline 4.1.5 kernel
(*) Both USB-md and KingCobra kernel modules are subsidiaries of VIRGO kernel
(*) There is a necessity for 64-bit version of USB-md for interoperability to VIRGO64 64-bit kernel on mainline version 4.13.3
(*) This requires separate repository for USB-md because of significant kernel function changes between 4.1.5 and 4.13.3 and idiosyncrasies of 64-bit
(*) USB-md driver has been rebuilt on 4.13.3 64-bit kernel after some changes to function prototypes and new usb-md64 repository is initialized with these commits

884. USBWWAN Kernel Log Spark Analyzer Update - Refactoring to a new python function - 18 June 2018

1. Spark Log Analyzer Spark_USBWWANLogMapReduceParser.py has been changed to modularize the pattern extraction by defining a new function accepting kern.log file, pattern and filter and also creates Spark DataFrame SQL table and queries it.
2. This is similar to NeuronRain AsFer log_mapreducer()

769. (FEATURE) USBWWAN analytics - USBmon and FTrace logs analysis - 15 November 2018 - this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

1. Logs Analysis for 2 standard kernel tracing facilities have been included - USBmon and FTrace. USBmon is the kernel debugfs tracing facility and FTrace is the Kernel functions tracing utility accessible from user space. (Kernel Address Sanitizer - KASAN - is only enabled in kernelspace via KBuild config and kernel build transparent to userspace)
2. USBmon traces are enabled by debugfs in /sys/kernel/debug/usb/usbmon and can be loaded in wireshark in libpcap format or usbmon pseudodevices can be viewed in tcpdump:

```
467 ls /sys/kernel/debug/
468 modprobe usbmon
472 dumpcap -D
474 ls /dev/usbmon0
475 ls -lrt /dev/usbmon*
487 tcpdump -i usbmon1
488 tcpdump -i usbmon2
489 tcpdump -i usbmon0
490 tcpdump -i usbmon3
491 tcpdump -i usbmon4
520 cat /sys/kernel/debug/usb/usbmon/1t 2>&1 > usbmon.mon
```

3. FTrace for function graph analysis are enabled by (Kernel.org FTrace Documentation:
https://www.kernel.org/doc/Documentation/trace/ftrace.txt):

```
536 ls /sys/kernel/debug/tracing/current_tracer
537 echo nop > /sys/kernel/debug/tracing/current_tracer
538 echo 0 > /sys/kernel/debug/tracing/tracing_on
539 echo $$ > /sys/kernel/debug/tracing/set_ftrace_pid
541 echo function > /sys/kernel/debug/tracing/current_tracer
545 echo 1 > /sys/kernel/debug/tracing/tracing_on
557 ls -lrt /sys/kernel/debug/tracing/trace
561 cat /sys/kernel/debug/tracing/set_graph_function
562 cat /sys/kernel/debug/tracing/trace_options
563 echo funcgraph-duration > /sys/kernel/debug/tracing/trace_options
566 cat /sys/kernel/debug/tracing/set_graph_function
567 cat /sys/kernel/debug/tracing/trace_options
568 cat /sys/kernel/debug/tracing/trace_options
569 echo funcgraph-cpu 2>&1 > /sys/kernel/debug/tracing/trace_options
620 cat /sys/kernel/debug/tracing/set_ftrace_pid
624 echo 7379 > /sys/kernel/debug/tracing/set_ftrace_pid
625 cat /sys/kernel/debug/tracing/trace 2>&1 > ftrace.log.15November2018
639 export JAVA_HOME=/media/Ubuntu2/jdk1.8.0_171/
640 export PATH=/usr/bin:$PATH
671 /media/Ubuntu2/spark-2.3.1-bin-hadoop2.7/bin/spark-submit Spark_USBWWANLogMapReduceParser.py 2>&1 >
testlogs/Spark_USBWWANLogMapReduceParser.FTraceAndUSBMon.log.15November2018
```

4. FTrace traces for specific userspace threads/processes are enabled by previous example commandlines and available through /sys/kernel/debug/tracing/trace (circular buffer). Function graph traces show kernel function invocations as call graph edges (denoted by fn2 <- fn1)

5. Spark_USBWWANLogMapReduceParser.py has been changed to invoke log analyzer for USBmon and FTrace logs for patterns Bi(BULK IN) and usb from USBmon and FTrace logs respectively:

- usbmon.15November2018.mon
- ftrace.ping.log.15November2018 (ftraces for ping of an IP address)

6. Logs for Spark Analyzer have been committed to Spark_USBWWANLogMapReduceParser.FTraceAndUSBMon.log.15November2018 which analyze the USBmon logs and WLAN traffic for IP address ping.

----

770. (THEORY and FEATURE) Program Analysis and Software Analytics - USBmd FTrace Kernel Function CallGraph Generation for Analysis - 22 November 2018 - this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

----

1.New bash shell script usb_md_ftrace.sh has been committed to repository which writes out an ftrace.log file containing kernel function call graph sequences for an executable code. It is invoked as:

```
        $usb_md_ftrace.sh <executable-to-trace>
```

usb_md_ftrace.sh summarizes previously mentioned ftrace options enabling commands into single file with an option for commandline argument of an executable to trace.

2.usb_wwan_modified/python-src/Spark_USBWWANLogMapReduceParser.py has been changed to include a new function ftrace_callgraph_dot() which parses an ftrace log generated by usb_md_ftrace.sh for command:

```
	$usb_md_ftrace.sh traceroute <ip-address>
```

3.ftrace_callgraph_dot() parses each line of ftrace.log and adds them as edges in a NetworkX Directed Graph. DOT file for this call graph is written to Spark_USBWWANLogMapReduceParser.ftrace_callgraph.dot

4.As a novelty, PageRank and Degree Centrality measures of the call graph NetworkX DiGraph are printed which show the prominently active regions of the kernel for traceroute . PageRank/Degree Centrality of kernel function callgraph is quite useful by treating every function caller as a voter to function callee. Theoretically, this centrality in kernel throws light on suspicious, malevolent invocations particularly involving memory and locking. In this traceroute ftrace example, lock and kmalloc functions have high centrality, and USB URB related functions are way down the ranking. More the ranking, deeper the function is in callstack trace in kernel.

5. Lot of functions have ISRA optimization of GCC. ISRA is known to cause signed int bugs (0 was erroneously promoted to 1 in loops) and ISRA has been disabled in ARM kernel: https://patchwork.kernel.org/patch/7113091/ by -fno-ipa-sra GCC flag. This kind of instability could be the reason for 32-bit VIRGO heisenbugs in string functions in older kernels.

6.Previous FTrace kernel call graph analysis is not only limited to USBmd WLAN analytics but can be applied to any executable requiring kernel profiling. Usual profilers measure time spent in the function whereas this graph theoretic analysis is superior and finds kernel bottlenecks and malicious patterns by analyzing call graphs within kernel.

7. Malicious code (e.g virus, worms, root-kits, bots, keystroke loggers) are usually associated with high cpu and memory footprint causing abnormal traffic. Analyzing infected kernel callgraph patterns might help in identifying the root cause.

8. FTrace kernel function call graph complements already implemented Program Analyzers: SATURN CFG driver in VIRGO kernels (accessible only in kernelspace) and Valgrind/KCachegrind/Callgrind userspace call graph analyzer in AsFer. By this kernel activity is partially visible and can be analyzed graph theoretically from userspace.

9. Outbreak of epidemics have been analyzed as Game Theoretic problem (https://blogs.cornell.edu/info2040/2016/09/16/game-theory-in-the-context-of-epidemics/) - on how people behave in epidemics and their conclusion - "faster information limits disease spread". Cybercrimes are epidemics counterpart in cloud of computers only difference being damage inflicted on intellectual property than humans and adversaries are hackers/malicious code in place of viri. This makes Cybercrimes a multi-player adversarial game involving Hackers/Malicious code Versus Aggrieved. Translating the previous conclusion to cybercrimes: Faster information about malicious code limits the damage.