```
    1: #include <linux/module.h> // included for all kernel modules
    2: #include <linux/kernel.h>    // included for KERN_INFO
    3: #include <linux/init.h>      // included for __init and __exit macros
    4: #include <linux/slab.h>
    5:
    6: //#include <stdio.h>
    7: //#include <stdlib.h>
    8:
    9: /*
   10: dead_compute01:
   11:
   12: This is used to create deadwrite/killing writes
   13: in two for loops.
   14:
   15: - given an array int *a and its size, write
   16:   new values to every a[i] in two for loops.
   17:   Between two for loops, do some reading in a.
   18: */
   19:
   20: int dead_compute01( int *a, int size){
   21:
   22:    int j=0 ;
   23:    long count=0;
   24:
   25:    for (j=0; j<size*3; j++){
   26:
   27:     int i, tmp=0;
   28:
   29:     // first loop to write a[i]s.
   30:     for (i = 0; i<size; i++){
   31:         a[i] = i*i-tmp;
   32:             tmp += i;
   33:     }
   34:
   35:     // do some reading
   36:     if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
   37:     printk(KERN_CONT "(j=%d)\t%d",j, a[(i+j)/4]);
   38:     if(j%10 == 0) printk(KERN_INFO "\n");
   39:
   40:     // second loop to write a[i]s
   41:     for (i =0 ; i< size; i++){
   42:         a[i] = i*i/2 - 1;
   43:                 tmp = tmp + 2*i -200;
   44:     }
   45:     count += tmp * 2 - tmp / 3;
   46:
   47:   }
   48:   return 0;
   49: }
   50:
   51: /*
   52: dead_compute02:
   53:
   54: This is used to create deadwrite/killing writes
   55: in two for loops. Similar to dead_compute01, but
   56: with no read between two write loops.
   57:
   58: - given an array int *a and its size, write
   59:   new values to every a[i] in two for loops.
   60: */
   61:
   62: int dead_compute02( int *a, int size){
   63:
   64:    int j=0 ;
   65:    long count=0;
   66:
   67:    for (j=0; j<size*3; j++){
   68:
   69:     int i, tmp=0;
   70:
```

```
   71:     // first loop to write a[i]s.
   72:     for (i = 0; i<size; i++){
   73:         a[i] = i*i-tmp;
   74:             tmp += i;
   75:     }
   76:
   77:     // second loop to write a[i]s
   78:     for (i =0 ; i< size; i++){
   79:         a[i] = i*i/2 + 1;
   80:                 tmp = tmp + 2*i -200;
   81:     }
   82:
   83:     count += tmp * 2 - tmp / 3;
   84:
   85:   }
   86:   return 0;
   87: }
   88:
   89:
   90: MODULE_LICENSE("GPL");
   91: MODULE_AUTHOR("LELE MA");
   92: MODULE_DESCRIPTION("A sample kernel module with simple deadwrites");
   93:
   94: static int __init k_array_test_init(void)
   95: {
   96:     int size = 2000;
   97:     int i;
   98:     int *a, *b;
   99:
  100:     a = kmalloc (size * sizeof(int), GFP_KERNEL);
  101:     b = kmalloc (size * sizeof(int), GFP_KERNEL);
  102:
  103:     printk(KERN_INFO "Starting k_array_test with deadwrites\n");
  104:
  105:     dead_compute01( a, size);
  106:
  107:     dead_compute02( b, size);
  108:
  109:     for (i = size-1; i>=0; i--){
  110:         printk(KERN_CONT "a[%d]=%d\t", i, a[i]);
  111:         printk(KERN_CONT "b[%d]=%d\t", i, b[i]);
  112:
  113:         if(i%10==0) printk(KERN_CONT "\n");
  114:     }
  115:
  116:     kfree(a);
  117:     kfree(b);
  118:     return 0;    // Non-zero return means that the module couldn't be loaded.
  119: }
  120:
  121: static void __exit k_array_test_cleanup(void)
  122: {
  123:     printk(KERN_INFO "Goodbye from k_array_test.\n");
  124: }
  125:
  126: module_init(k_array_test_init);
  127: module_exit(k_array_test_cleanup);
  128:
```