```
    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:
   14: in two for loops.
   15:
   16: - given an array int *a and its size, write
   17:
   18:   new values to every a[i] in two for loops.
   19:
   20:   Between two for loops, do some reading in a.
   21:
   22: - int *a
   23: - int size
   24:
   25: */
   26:
   27: int dead_compute01( int *a, int size){
   28:
   29:    int j=0 ;
   30:    long count=0;
   31:
   32:    for (j=0; j<size*3; j++){
   33:
   34:     int i, tmp=0;
   35:
   36:     // first loop to write a[i]s.
   37:     for (i = 0; i<size; i++){
   38:         a[i] = i*i-tmp;
   39:             tmp += i;
   40:     }
   41:
   42:     // do some reading
   43:     if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
   44:     printk(KERN_CONT "(j=%d)\t%d",j, a[(i+j)/4]);
   45:     if(j%10 == 0) printk(KERN_INFO "\n");
   46:
   47:     // second loop to write a[i]s
   48:     for (i =0 ; i< size; i++){
   49:         a[i] = i*i/2 - 1;
   50:                 tmp = tmp + 2*i -200;
   51:     }
   52:
   53:     count += tmp * 2 - tmp / 3;
   54:
   55:     }
   56:     return 0;
   57: }
   58:
   59: /*
   60: dead_compute02:
   61:
   62: This is used to create deadwrite/killing writes
   63:
   64: in two for loops. Similar to dead_compute01, but
   65:
   66: with no read between two write loops.
   67:
   68: - given an array int *a and its size, write
   69:
   70:   new values to every a[i] in two for loops.
   71:
   72: - int *a
   73: - int size
   74:
   75: */
   76:
   77: int dead_compute02( int *a, int size){
```

```
 78:
 79:    int j=0 ;
 80:    long count=0;
 81:
 82:    for (j=0; j<size*3; j++){
 83:
 84:     int i, tmp=0;
 85:
 86:     // first loop to write a[i]s.
 87:     for (i = 0; i<size; i++){
 88:         a[i] = i*i-tmp;
 89:             tmp += i;
 90:     }
 91:
 92:     // second loop to write a[i]s
 93:     for (i =0 ; i< size; i++){
 94:         a[i] = i*i/2 + 1;
 95:                 tmp = tmp + 2*i -200;
 96:     }
 97:
 98:     count += tmp * 2 - tmp / 3;
 99:
100:     }
101:     return 0;
102: }
103:
104:
105: MODULE_LICENSE("GPL");
106: MODULE_AUTHOR("LELE MA");
107: MODULE_DESCRIPTION("A sample kernel module with simple deadwrites");
108:
109: static int __init k_array_test_init(void)
110: {
111:     int size = 2000;
112:     int i;
113:     int *a, *b;
114:
115:     a = kmalloc (size * sizeof(int), GFP_KERNEL);
116:     b = kmalloc (size * sizeof(int), GFP_KERNEL);
117:
118:     printk(KERN_INFO "Starting k_array_test with deadwrites\n");
119:
120:     dead_compute01( a, size);
121:
122:     dead_compute02( b, size);
123:
124:     for (i = size-1; i>=0; i--){
125:         printk(KERN_CONT "a[%d]=%d\t", i, a[i]);
126:         printk(KERN_CONT "b[%d]=%d\t", i, b[i]);
127:
128:         if(i%10==0) printk(KERN_CONT "\n");
129:     }
130:
131:     kfree(a);
132:     kfree(b);
133:     return 0;    // Non-zero return means that the module couldn't be loaded.
134: }
135:
136: static void __exit k_array_test_cleanup(void)
137: {
138:     printk(KERN_INFO "Goodbye from k_array_test.\n");
139: }
140:
141: module_init(k_array_test_init);
142: module_exit(k_array_test_cleanup);
143:
144: //int init_module(void){
145: //  printk(KERN_INFO "HELLO WORLD!\n");
146: //
147: //  return 0;
148: //}
149:
150: //void cleanup_module(void)
151: //{
152: //  printk(KERN_INFO "Goodbye from k_array_test.\n");
153: //}
154:
```