T1.
Bug Hunter's Diaryssa löydettiin haavoittuvuus VLC ohjelmasta.
Haavoittuvuus löydettiin seuraamalla (TiVo muotoisen) video tiedoston lukua lähdekoodissa.
Kirjoittaja seurasi datan erottelevia funktioita (demux) ja etsi ohjelmointi virheitä.
Hän löysi mahdollisen Ylivuoto ongelman, jota pystyi hyödyntämään muokkaamalla
TiVo tiedostoa, kaataen VLC ohjelman, ja saaden siten kontrollin EIP rekisteristä.
Tämä mahdollistaa minkä tahansa ohjeiden suorittamisen koneella. Jos sitten VLC:tä
käytetään esim. internetin kautta, on mahdollista ajaa mitä tahansa koodia serverillä.

Kyseessä on injektio hyökkäys. Se voi toimia monella tapaa, mutta kaikille on yhteistä se,
että jonkunlaista omaa koodia ajetaan paikassa, jossa sen ei pitäisi olla mahdollista.

Hyökkäys mahdollistaa paljon, riippuen miten haavoittuvuus on saavutettu, ja missä.
Voidaan mm. muokata nettisivun toimintaa, kopioida yksityisiä tietoja, tai muuten vaan
riehua Serverillä.

T2.

```
// No leaks found with gdb, I suppose it would be possible though

/* With valgrind:
LEAK SUMMARY:
==24114== definitely lost: 400 bytes in 1 blocks
==24114== indirectly lost: 0 bytes in 0 blocks
==24114== possibly lost: 0 bytes in 0 blocks
==24114== still reachable: 0 bytes in 0 blocks
==24114== suppressed: 0 bytes in 0 blocks
*/

/* The program isn't correct, since the array has 100 elements, hence the last element has
index 99, but we are setting a value at index 100.
The value is set, but it is not safe and doesn't actually belong to the array.
*/
```

Koodi liittenä (memleak.c)

T3.

Part 1:
00000000000000000000000000000000000000000000000010000000000000000000000000000000
00000000000000000000
50th after freeing: 1
After freeing, the program runs, and the values remain, but the values are marked as 'free'
Valgrind: All heap blocks were freed -- no leaks are possible
Now running 'free(&p_arr[50]);' is not possible, as the pointer type is incompatible (this address is in heap memory? Atleast it is not allocated with malloc/etc.)
Output would be:
free(): invalid pointer
Aborted (core dumped)

Part 2:
Creating a vector with malloc, and appending values to it with realloc.
After initialization: 1234
Added 9: 12349
Added 9: 123499
Added 9: 1234999
This implementation is not really useful, as it is slow to realloc each time a value is added.
Instead, most C implementations of dynamic arrays / vectors double the capacity of the vector with realloc, when capacity is exceeded.
Linked lists and dynamic arrays serve different purposes, to summarize:
Linked lists provide O(1) insertion and deletion, due to the linked structure, but O(n) when referencing, due to traversing the list.
Vectors on the other hand provide O(1) referencing, due to contiguous blocks of memory, but O(n) insertion and deletion, due to copying and allocating memory.

Koodi liittenä (T3.c)