**Task 1**

**a)**

|  |
| --- |
| kali@kali:~$ for run in {1..10}; do echo "Fuzztest 1337" | radamsa ; done  Guzztest 2  Fuzztest 1336  Nuzztest 1ʷ-86  Fuzztest 1  Fuzztest �1210  Fuzztest 1  Fuzztest 1337  Fuzztest 1336  Fuzztest 1337  Fuzztest 71046994280470294509227 |

**b)**

|  |
| --- |
| for i in {1..100}; do cat a.txt | radamsa |cat > fuzz$i.txt ; done |

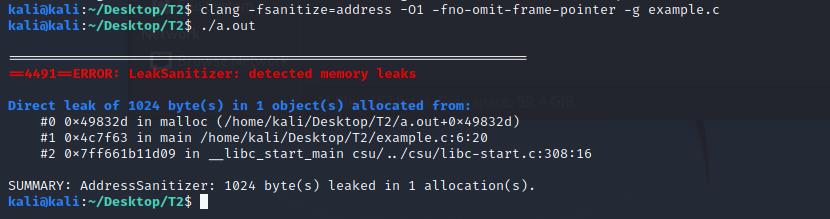
**Task 2**

**a)**

**Command line used to compile the program**

|  |
| --- |
| clang -fsanitize=address -O1 -fno-omit-frame-pointer -g example.c |
|  |

**Screenshot of the result after running the program**



**What is the error and what is causing it in this program?**

Memoy leak, memory allocated with malloc() but not freed with free()

**b)**

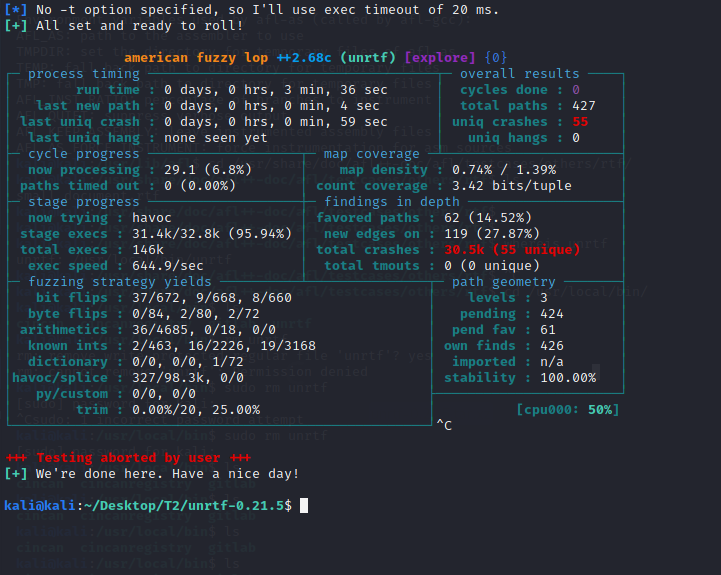
**Command line used to configure unrtf**

|  |
| --- |
| sudo ./configure CC="/usr/bin/afl-gcc" --prefix=$HOME/unrt  make  make install  /unrt/bin/unrtf --html /usr/share/doc/afl++-doc/afl/testcases/others/rtf/small\_document.rtfunrt/bin/unrtf --html /usr/share/doc/afl++-doc/afl/testcases/others/rtf/small\_document.rtf  mkdir input  mkdir output  cp /usr/share/doc/afl++-doc/afl/testcases/others/rtf/small\_document.rtf ./input/ |

**Command line used to run AFL**

|  |
| --- |
| fl-fuzz -i ./input/ -o ./output/ ~/unrt/bin/unrtf |

**Screenshot of the AFL status screen after stopping the fuzzer**



**What do you think are the most significant pieces of information on the status screen? Why are they important?**

**Uniq Crashes: How many different unique crash scenarios there are, implicates unique errors in code.**

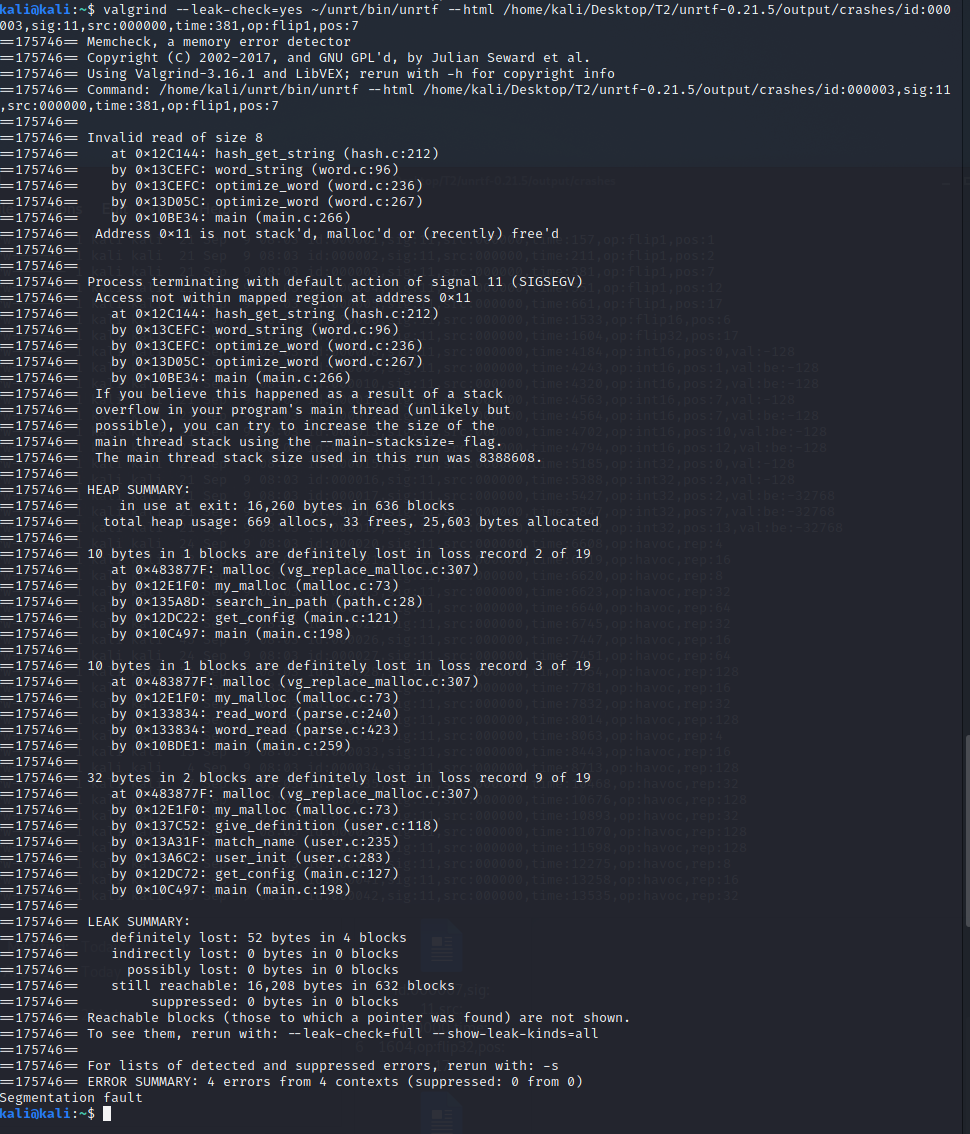
**Total crashes: This with the run time gives valuable information about stability.**

**run time: ^same as above.**

**last uniq crash: Determines when “enough” fuzzing has been done for now (for this version of rolling release for example).**

**c)**

**Take a screenshot of the Valgrind result after running the program**



**What can you tell about the crash?**

It was a memory error, some memory address space was not allocated or freed.

**Task3**

**Provide the C-code of your program**

|  |
| --- |
| **#include <stdio.h>**  **#include <ctype.h>**  **int main(int argc, char \*\*argv){**  **char c;**  **int i;**  **FILE \*f=fopen(argv[1],"r"); // read file**  **int ok =1 ;**  **while(ok && i != EOF){**  **i=fgetc(f);**  **ok=isdigit(i);**  **}**  **if( i == ' '){**  **ok = 1;**  **while(ok && c != EOF && c != '\n'){**  **c=fgetc(f);**  **ok=isalpha(c);**  **}**  **if(c=='\n'){c=fgetc(f);}**  **if( c ==EOF ){**  **return 1;**  **}**  **return 0;**  **}**  **return 0;**  **}** |

**Take a screenshot of the AddressSanitizer results after running your program with the test cases. Show at least 3 ASAN outputs.**

