Laborator 3 – Rezolvare

Pentru a rula acest laborator, este necesară o mașină virtuală, care să nu aibă antivirus pe ea (sau propriul calculator, dar fără antivirus pornit).

Prerequisites: *VMware Workstation 16 Player* & *Ubuntu 64-bit 20.04 LTS*

**Ex 1**. *Please create a new folder, lab3, in your home directory. Change the sample provided in the archive, to allow you to count how many times was accessed each file from lab3 folder. Please create another folder in the lab3 folder, lets call it lab31 and see if your application is counting how many times were accesssed each file from lab31 folder. Please explain the obtained results.*

/\* Create, Delete & Count Access files in lab3, respectevly lab3->lab31 \*/

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/types.h>

#include <sys/inotify.h>

#include <unistd.h>

#include <string.h>

#define EVENT\_SIZE ( sizeof (struct inotify\_event) )

#define EVENT\_BUF\_LEN ( 1024 \* ( EVENT\_SIZE + 16 ) )

// Structura în care păstrăm nr de câte ori a fost modificat/accesat fișierul

struct filename\_counter

{

char filename[50];

int nr;

} v[30];

int main()

{

int length, i = 0, last\_file = 0;

int fd;

int wd;

char buffer[EVENT\_BUF\_LEN];

/\* creating the INOTIFY instance \*/

fd = inotify\_init();

/\* checking for error \*/

if(fd < 0) perror( "inotify\_init" );

/\* adding the directory into watch list. Here, the suggestion is to validate the existence of the directory before adding into monitoring list. \*/

wd = inotify\_add\_watch(fd, "/mnt/c/users/Larisa/Desktop/lab3", IN\_CREATE | IN\_DELETE | IN\_ACCESS);

/\* read to determine the event change happens on the directory. Actually this reads blocks until the change event occurs \*/

do

{

i = 0;

length = read(fd, buffer, EVENT\_BUF\_LEN);

/\* checking for error \*/

if(length < 0) perror( "read" );

/\* actually read return the list of change events happens. Here, read the change event one by one and process it accordingly. \*/

while(i < length)

{

struct inotify\_event \*event = (struct inotify\_event\*) &buffer[i];

if(event->len)

{

if(event->mask & IN\_CREATE)

{

if(event->mask & IN\_ISDIR)

printf("New directory %s created.\n", event->name);

else

printf("New file %s created.\n", event->name);

}

else if(event->mask & IN\_DELETE)

{

if(event->mask & IN\_ISDIR)

printf("Directory %s deleted.\n", event->name);

else

printf("File %s deleted.\n", event->name);

}

else if(event->mask & IN\_ACCESS)

{

int j, found\_file = 0;

// căutăm în tot vectorul să vedem dacă fișierul a mai fost accesat sau e la prima accesare

for(j = 0; j < 30; j++)

{

// dacă fișierul a fost accesat

if(strcmp(v[j].filename, event->name) == 0)

{

v[j].nr++;

found\_file = 1; // am marcat faptul că am găsit fișierul

break;

}

}

// dacă fișierul nu a mai fost accesat până acum

if(found\_file == 0)

{

v[last\_file + 1].nr = 1; // marcăm faptul că a fost modificat/accesat o dată

strcpy(v[last\_file + 1].filename, event->name); // punem și numele fișierului

last\_file++; // ultima celulă care are date, ca să știu unde pun noile adăugiri

}

if (event->mask & IN\_ISDIR)

if(j < 30)

printf( "Directory %s accessed (modified) %d times.\n", event->name, v[j].nr);

else

printf( "Directory %s accessed (modified) %d times.\n", event->name, v[last\_file].nr);

else

if(j < 30)

printf("File %s accessed (modified) %d times.\n", event->name, v[j].nr);

else

printf("File %s accessed (modified) %d times.\n", event->name, v[last\_file].nr);

}

}

i += EVENT\_SIZE + event->len;

}

} while(1);

/\* removing thedirectory from the watch list. \*/

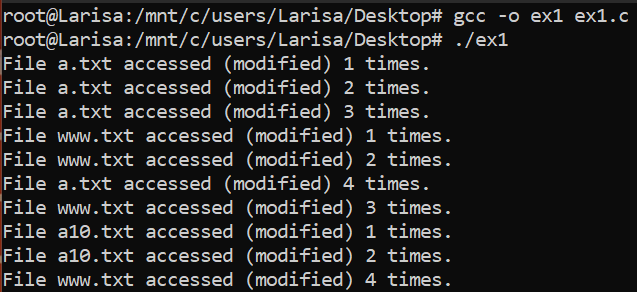
inotify\_rm\_watch(fd, wd);

/\*closing the INOTIFY instance\*/

close(fd);

}

Aplicația nu numără de câte ori a fost accesat fiecare fișier din lab31, deoarece atunci când un director este monitorizat, *inotify* va returna evenimente pentru directorul însuși și pentru fișierele/directoarele din interiorul directorului, DAR nu și pentru fișierele din subdirectoarele directorului principal (cel monitorizat). Deci va apărea că lab31 a fost accesat, dar nu va apărea nimic legat de fișierele din acesta.



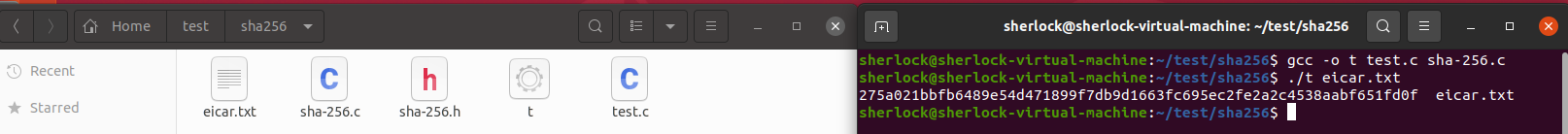
([Sursă1](https://developer.ibm.com/tutorials/l-inotify/), [Sursă2](https://man7.org/linux/man-pages/man1/inotifywatch.1.html), [Sursă3](https://sidmartin-bio.com/is-inotify-recursive/))

**Ex 2**. *Please compute the sha256 hash for eicar.txt file. Now enter on this site* [*https://www.virustotal.com/gui/home/upload*](https://www.virustotal.com/gui/home/upload) *and select search tab. Please insert the computed sha256 hash and see the results you obtain for that hash. Try similar tests with other files and see the results.*

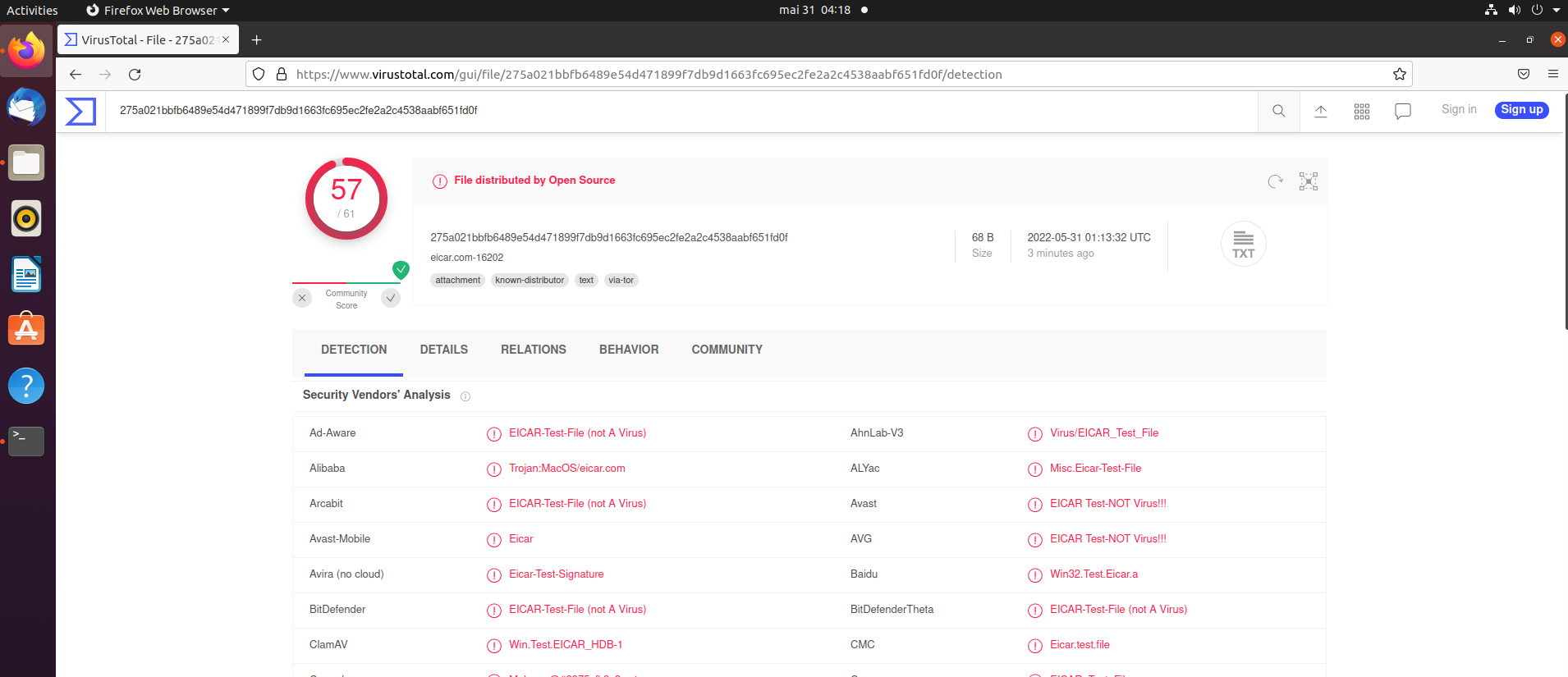
Comenzi: ***gcc -o t test.c sha-256.c***

***./t eicar.txt*** → obținem:

*275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f* eicar.txt



Accesând site-ul și inserând *hash-ul* în searchBar, obținem (*57/61 vendors flagged this file as malicious*):



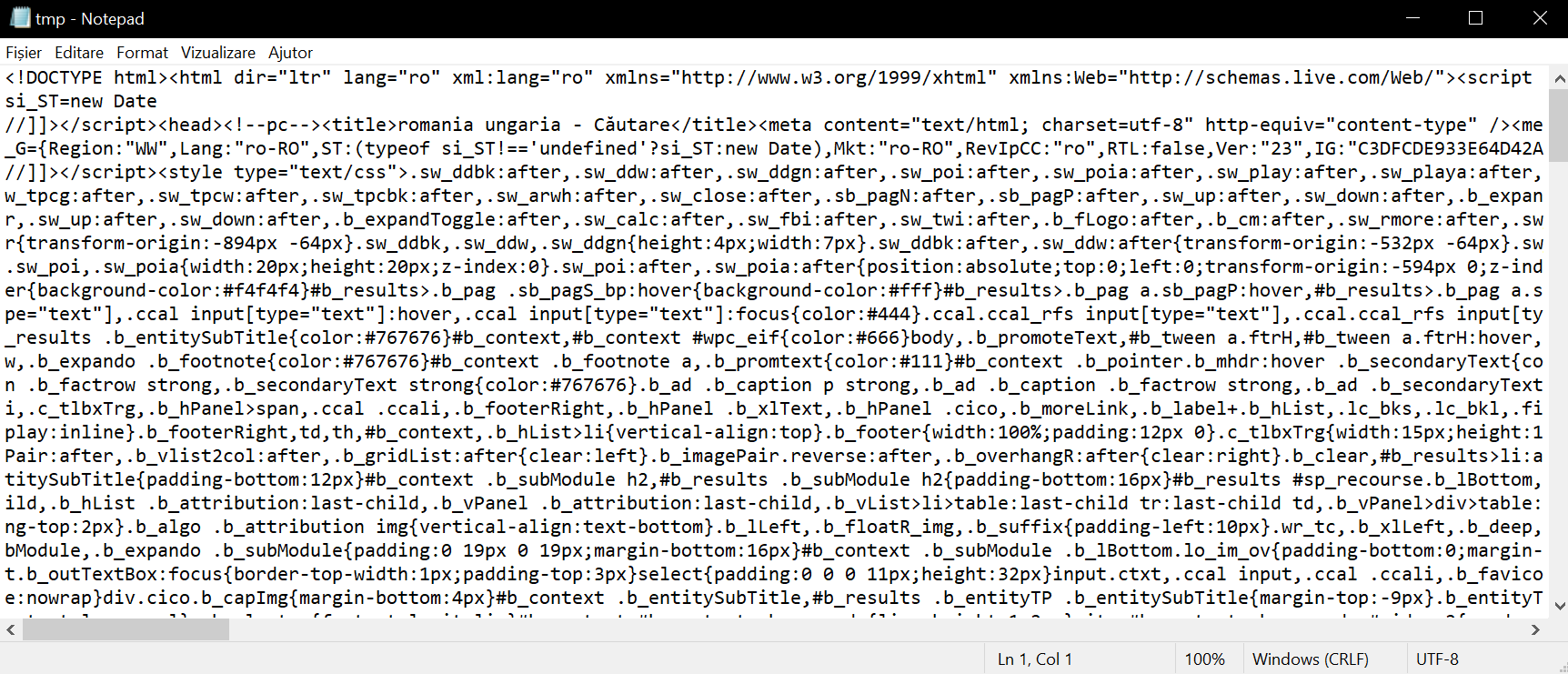
**Ex 3**. *In folder curl from the archive you will find a small sample which uses curl to send a http(s) request to a server and to get the server response. Please look at the code. Change the requested url with different other urls you are usually accessing and see if you obtain those pages.*

Comenzi: ***sudo apt-get install curl***

***sudo apt-get install libcurl4-openssl-dev***

***gcc curl.c -o curl -O2 -Wall -lcurl*** → O2 (cod optimizat); Wall (afișează avertismente mai detaliate); curl (leagă programul de *libcurl*)

***./curl***

Vom obține fișierul *tmp.txt*, cu următorul conținut:

Obținem rezultate similare și pentru <https://www.google.com> sau pentru https://www. youtube.com/?gl=RO .

**Ex 4**. *Using the sample you have implemented with inotify please monitor a folder and for every created or accessed file do compute its sha256 hash and using virus total api* [*https://developers.virustotal.com/reference*](https://developers.virustotal.com/reference) *and curl send a request to virus total api to see what it knows about the files computed hashes. If you obtain the answer the hash correspond to an infected file, change your code to delete the file.*

Modificăm *Ex1.c* pentru a implementa noile cerințe.

/\* Monitorizăm fișierul lab3. Pt fiecare fișier creat, accesat sau modificat, verificăm cu API să nu fie virus.

Dacă este virus, îl ștergem. \*/

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/types.h>

#include <sys/inotify.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <curl/curl.h>

#include <openssl/md5.h>

#define EVENT\_SIZE ( sizeof (struct inotify\_event) )

#define EVENT\_BUF\_LEN ( 1024 \* ( EVENT\_SIZE + 16 ) )

#define MAXCHAR 10000

unsigned char result[MD5\_DIGEST\_LENGTH];

// Get the size of the file by its file descriptor

unsigned long get\_size\_by\_fd(int fd) {

struct stat statbuf;

if(fstat(fd, &statbuf) < 0) exit(-1);

return statbuf.st\_size;

}

unsigned char\* computeSHA\_file(char \*file)

{

int file\_descript;

unsigned long file\_size;

char\* file\_buffer;

file\_descript = open(file, O\_RDONLY);

if(file\_descript < 0) exit(-1);

file\_size = get\_size\_by\_fd(file\_descript);

file\_buffer = mmap(0, file\_size, PROT\_READ, MAP\_SHARED, file\_descript, 0);

MD5((unsigned char\*) file\_buffer, file\_size, result);

munmap(file\_buffer, file\_size);

return result;

}

size\_t write\_callback(char \*ptr, size\_t size, size\_t nmemb, void \*userdata)

{

return fwrite(ptr, size, nmemb, (FILE \*) userdata);

}

int main()

{

int length, i = 0, last\_file = 0;

int fd;

int wd;

char buffer[EVENT\_BUF\_LEN];

char working\_directory[] = "/mnt/c/users/Larisa/Desktop/lab3";

CURL \*curl;

CURLcode res;

FILE \* f = NULL;

char response\_buffer[MAXCHAR];

const char\* base\_url = "https://www.virustotal.com/vtapi/v2/file/report?apikey=";

//my API key

const char\* api\_key = "1e5c8a20708ee442ef7393f7d7ed4c714137d2dd3ee7d9dcccf5c5eaeeac5127";

const char\* resourse\_string = "&resource=";

const int hash\_length = 64;

/\* creating the INOTIFY instance \*/

fd = inotify\_init();

/\* checking for error \*/

if(fd < 0) perror( "inotify\_init" );

/\* adding the directory into watch list. Here, the suggestion is to validate the existence of the directory before adding into monitoring list. \*/

wd = inotify\_add\_watch(fd, working\_directory, IN\_CREATE | IN\_ACCESS | IN\_MODIFY);

/\* read to determine the event change happens on the directory. Actually this reads blocks until the change event occurs \*/

do

{

i = 0;

length = read(fd, buffer, EVENT\_BUF\_LEN);

/\* checking for error \*/

if(length < 0) perror( "read" );

/\* actually read return the list of change events happens. Here, read the change event one by one and process it accordingly. \*/

while(i < length)

{

struct inotify\_event \*event = (struct inotify\_event\*) &buffer[i];

if(event->len)

{

if(event->mask & IN\_MODIFY) // Dacă se crează un nou fișier, automat intră în IN\_MODIFY (nu mai are sens să avem un if cu IN\_CREATE)

{

// Determine the absolute path of the file

char \* filename = (char\*) malloc(strlen(working\_directory) + strlen(event->name) + 2);

strcpy(filename, working\_directory);

strcat(filename, "/");

strcat(filename, event->name);

// compute hash

unsigned char\* hash = computeSHA\_file(filename);

char total[1000];

for(int i = 0; i < MD5\_DIGEST\_LENGTH; i++) {

sprintf(&total[i\*2], "%02x", (unsigned int)hash[i]);

}

// Generate the url to the virus API

char \*url = (char\*) malloc(strlen(base\_url) + strlen(api\_key) + strlen(resourse\_string) + strlen(total) + 1);

strcpy(url, base\_url);

strcat(url, api\_key);

strcat(url, resourse\_string);

strcat(url, total);

// Sent the request to the virus total API

curl = curl\_easy\_init();

if(curl) {

curl\_easy\_setopt(curl, CURLOPT\_URL, url);

f = fopen("tmp.txt", "w+");

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, f);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_callback);

/\* Perform the request, res will get the return code \*/

res = curl\_easy\_perform(curl);

/\* Check for errors \*/

if(res != CURLE\_OK)

fprintf(stderr, "curl\_easy\_perform() failed: %s\n",

curl\_easy\_strerror(res));

fclose(f);

/\* always cleanup \*/

curl\_easy\_cleanup(curl);

}

// Read the output from the API and search it for the number of positive detections.

// If this number is bigger than 0, then the file will be removed.

if(!(f = fopen( "tmp.txt", "r")))

{

perror( "fopen" );

return( 1 );

}

while (fgets(response\_buffer, MAXCHAR-1, f) != NULL) {

int no\_positives;

char\* positive\_ptr = strstr(response\_buffer, "positives");

if(positive\_ptr != NULL) {

positive\_ptr = positive\_ptr + strlen("positives\": ");

positive\_ptr = strtok(positive\_ptr, ",");

no\_positives = atoi(positive\_ptr);

if(no\_positives > 0) {

if (remove(filename) == 0)

printf("Deleted successfully the file %s.\n", filename);

else

printf("Unable to delete the file %s.\n", filename);

}

}

}

fclose(f);

free(filename);

free(url);

}

else if(event->mask & IN\_ACCESS)

{

// Determine the absolute path of the file

char \* filename = (char\*) malloc(strlen(working\_directory) + strlen(event->name) + 2);

strcpy(filename, working\_directory);

strcat(filename, "/");

strcat(filename, event->name);

// compute hash

unsigned char\* hash = computeSHA\_file(filename);

char total[1000];

for(int i = 0; i < MD5\_DIGEST\_LENGTH; i++) {

sprintf(&total[i\*2], "%02x", (unsigned int)hash[i]);

}

// Generate the url to the virus API

char \*url = (char\*) malloc(strlen(base\_url) + strlen(api\_key) + strlen(resourse\_string) + strlen(total) + 1);

strcpy(url, base\_url);

strcat(url, api\_key);

strcat(url, resourse\_string);

strcat(url, total);

// Sent the request to the virus total API

curl = curl\_easy\_init();

if(curl) {

curl\_easy\_setopt(curl, CURLOPT\_URL, url);

f = fopen("tmp.txt", "w+");

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, f);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_callback);

/\* Perform the request, res will get the return code \*/

res = curl\_easy\_perform(curl);

/\* Check for errors \*/

if(res != CURLE\_OK)

fprintf(stderr, "curl\_easy\_perform() failed: %s\n",

curl\_easy\_strerror(res));

fclose(f);

/\* always cleanup \*/

curl\_easy\_cleanup(curl);

}

// Read the output from the API and search it for the number of positive detections.

// If this number is bigger than 0, then the file will be removed.

if(!(f = fopen( "tmp.txt", "r")))

{

perror( "fopen" );

return( 1 );

}

while (fgets(response\_buffer, MAXCHAR-1, f) != NULL) {

int no\_positives;

char\* positive\_ptr = strstr(response\_buffer, "positives");

if(positive\_ptr != NULL) {

positive\_ptr = positive\_ptr + strlen("positives\": ");

positive\_ptr = strtok(positive\_ptr, ",");

no\_positives = atoi(positive\_ptr);

if(no\_positives > 0) {

if (remove(filename) == 0)

printf("Deleted successfully the file %s.\n", filename);

else

printf("Unable to delete the file %s.\n", filename);

}

}

}

fclose(f);

free(filename);

free(url);

}

}

i += EVENT\_SIZE + event->len;

}

} while(1);

/\* removing thedirectory from the watch list. \*/

inotify\_rm\_watch(fd, wd);

/\*closing the INOTIFY instance\*/

close(fd);

}

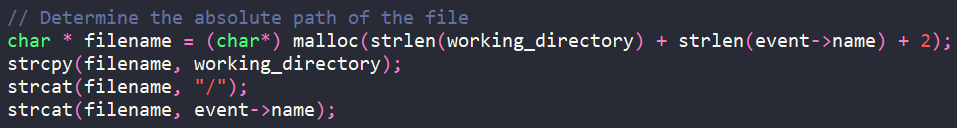
Comandă: ***gcc -o ex4 Ex4.c -lcurl -lssl -lcrypto && ./ex4***

Pe pași, explicația programului:

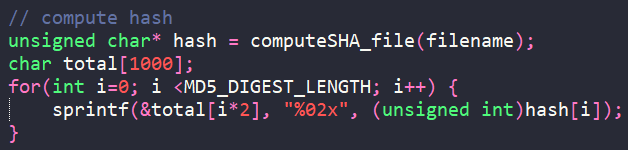
Ne alegem directorul care va fi urmărit:



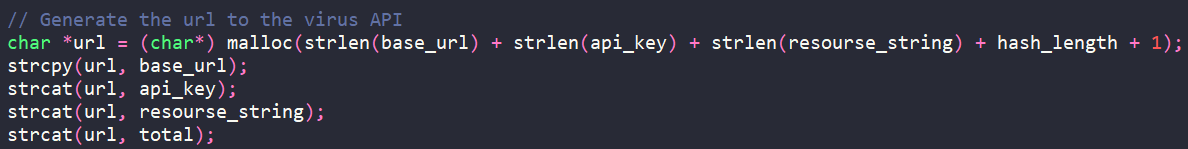
Dacă este creat sau accesat un fișier, trebuie să îi obținem numele:



Pentru a putea să obținem hash-ul acestuia:

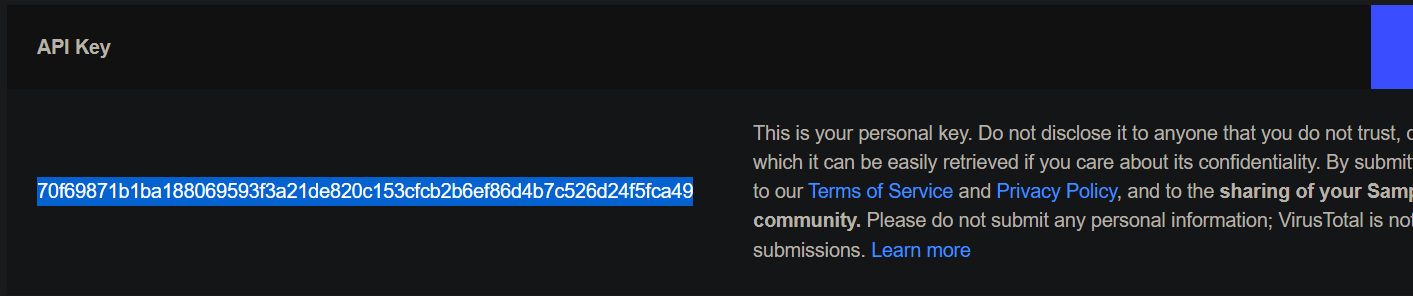


Trebuie să generăm API-ul corespunzător pentru a putea scana acest hash. Trebuie să ne creăm cont pe <https://www.virustotal.com/> și vom obține o cheie API. Pașii și documentația sunt [aici](https://www.tines.com/blog/virustotal-api-security-automation).

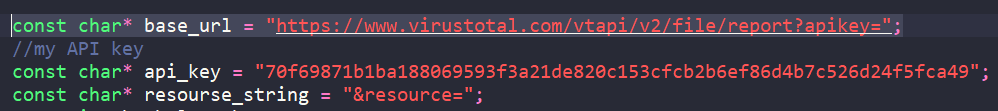


Spre exemplu, API-ul pentru hash-ul de ECARI.txt ar fi:

*https://www.virustotal.com/vtapi/v2/file/report?apikey=70f69871b1ba188069593f3a21de820c153cfcb2b6ef86d4b7c526d24f5fca49&resource=44d88612fea8a8f36de82e1278abb02f*



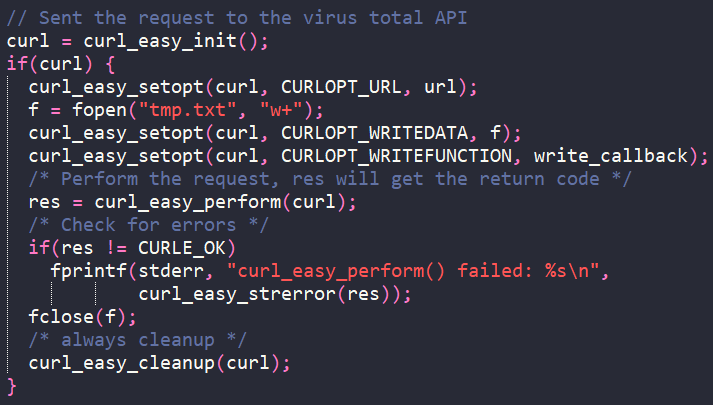
Împărțire:



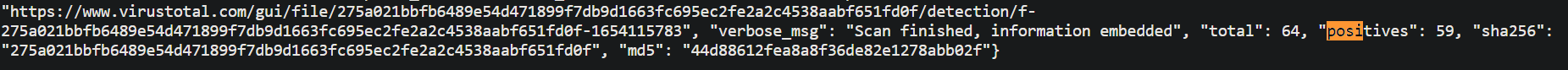
La care se adaugă și hash-ul fișierului. Și dacă introducem acest URL, vom obține:

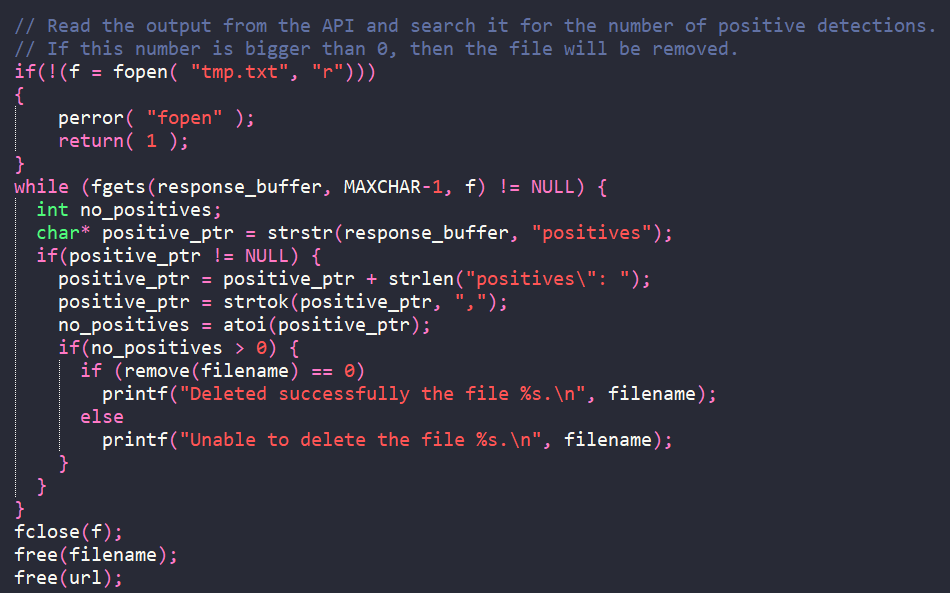


Deci, folosim ***CURL***.



Și dacă numărul de *positives* din fișier e mai mare de 0, ștergem fișierul. Pentru ECARI.txt, *positives*:





În terminal:

