Centro Universitário Senac

Alunos:

Higor Viana de Morais

Josué Batista Matos Deschamps de Melo

Ygor Kaique de Souza Pinto

Relatório do EP: Fuse e Esteganografia

Professor: Eduardo Heredia

Curso: Bacharelado em Ciência da Computação

Disciplina: Sistemas Operacionais

Instituição: Centro Universitário Senac Santo Amaro

São Paulo

2020

Introdução

Neste projeto o desafio proposto era de se realizar o uso de Fuse (Filesystem in Userspace) e Esteganografia, que é uma forma de segurança por obscurantismo implementado com a linguagem C.

Realizamos um estudo no Github de Sadimanna para entender como funciona a manipulação de imagens em Bitmap de 16 cores, ajudando na implementação da esteganografia, e ainda com as referências que o Professor Eduardo Heredia passou durante o semestre.

Para que seja possível a manipulação de imagens, usamos imagens de Bitmap de 16 cores para conseguir fazer a manipulação do arquivo, com o uso de arquivo em txt para ser colocado dentro da imagem.

Em nosso projeto utilizamos o sistema operacional Ubuntu na versão 20.04.1, o Git para fazer a interação do grupo e o VSCode como editor de texto para programar o código e executar todos os procedimentos envolvidos.

Desenvolvimento

Iniciamos o projeto com a parte da esteganografia, pois tivemos mais facilidade de implementar. Na sequência consultamos o material de aula e fizemos pesquisas relacionadas ao FUSE. Porém tivemos dificuldade para juntar parte da esteganografia com a parte do FUSE. Explicando algumas funções:

get_imagem_para_deslocamento(): Captura a imagem para deslocamento e faz uma conversão para inteiro, fazendo ser possível o uso do binário

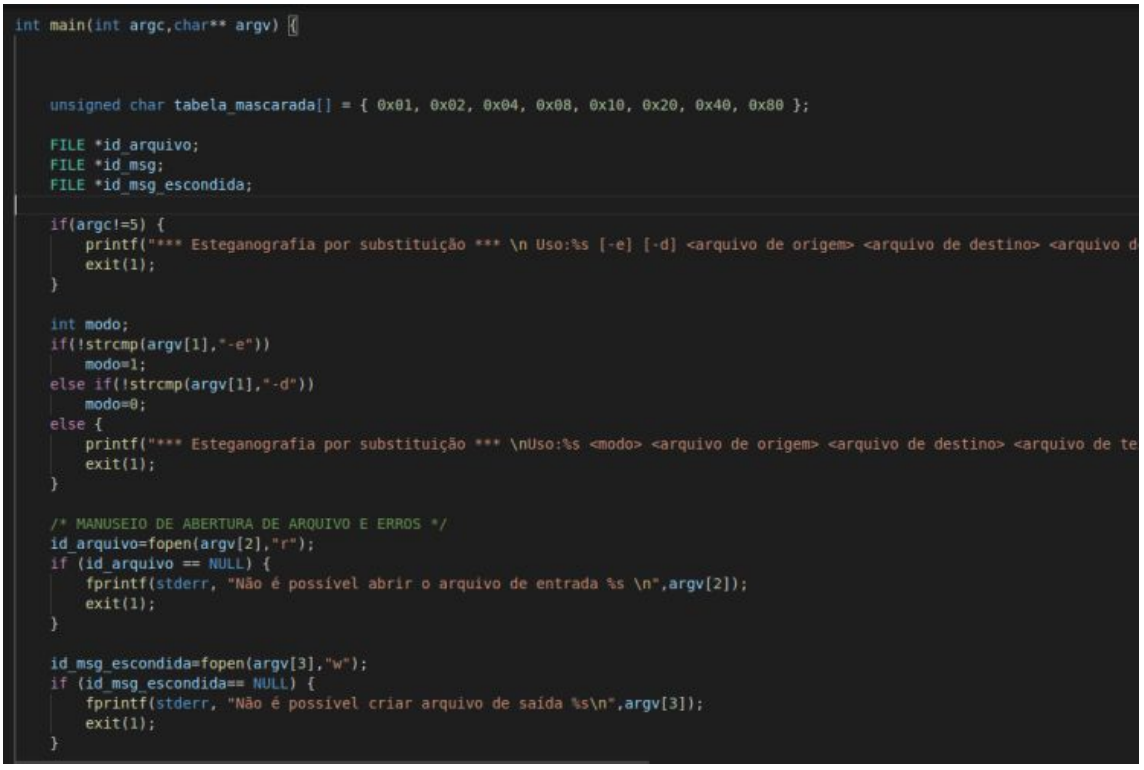
get_mensagem_length(): Configura o tamanho da mensagem.

get_bit(): Captura cada bit da imagem para leitura dos bits.

```
//************************
int get_imagem_para_deslocamento(FILE* bmp_deslocamento) {
    fseek(bmp_deslocamento,10,0);
    int deslocamento;
    deslocamento=(int)fgetc(bmp_deslocamento);
    return deslocamento;
}

int get_mensagem_length(FILE *fp) {
    fseek(fp, 0L, SEEK_END);
    int tamanho = ftell(fp);
    fseek(fp, 0L, SEEK_SET);
    return(tamanho);
}
int get_bit(char o_byte,int outro_bit) {
    return((o_byte>>8-outro_bit)&1);
}
```

tabela_mascara[]: Serve para converter a imagem em binário e fazer a esteganografia.
id_arquivo: Identificador do Arquivo, para saber qual arquivo está sendo manipulado durante a execução do código.
id_msg: Identificador da mensagem que está sendo passada para se tornar uma mensagem escondida.
id_msg_escondida: Identificador da mensagem que foi escondida.
Algumas passagens de argumentos são tratadas com if para assim fazer uma espécie de contorno com possíveis exceções na execução da aplicação.



```c
int main(int argc,char** argv) {



    unsigned char tabela_mascarada[] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

    FILE *id_arquivo;
    FILE *id_msg;
    FILE *id_msg_escondida;

    if(argc!=5) {
        printf("*** Esteganografia por substituição *** \n Uso:%s [-e] [-d] <arquivo de origem> <arquivo de destino> <arquivo d
        exit(1);
    }

    int modo;
    if(!strcmp(argv[1],"-e"))
        modo=1;
    else if(!strcmp(argv[1],"-d"))
        modo=0;
    else {
        printf("*** Esteganografia por substituição *** \nUso:%s <modo> <arquivo de origem> <arquivo de destino> <arquivo de te
        exit(1);
    }

    /* MANUSEIO DE ABERTURA DE ARQUIVO E ERROS */
    id_arquivo=fopen(argv[2],"r");
    if (id_arquivo == NULL) {
        fprintf(stderr, "Não é possível abrir o arquivo de entrada %s \n",argv[2]);
        exit(1);
    }

    id_msg_escondida=fopen(argv[3],"w");
    if (id_msg_escondida== NULL) {
        fprintf(stderr, "Não é possível criar arquivo de saida %s\n",argv[3]);
        exit(1);
    }
```

arq_buffer: variável temporária para um byte do arquivo

msg_buffer: buffer temporário para um byte de mensagem

```c
int c=0;

/* Gera arquivo com o mesmo cabeçalho. Copie os primeiros 128 bytes */
char tmp_sig_cpy;
int deslocamento=get_imagem_para_deslocamento(id_arquivo);

rewind(id_arquivo);

for(int i=0;i<deslocamento;i++) {
    tmp_sig_cpy=fgetc(id_arquivo);
    fputc(tmp_sig_cpy,id_msg_escondida);
    c++;
}
/*Arquivo criado como .bmp */

char arq_buffer;            // Variável temporária para um byte do arquivo
char msg_buffer;        // Buffer temporário para um byte de mensagem

if(modo) {
    id_msg=fopen(argv[4],"r");
    if (id_msg== NULL) {
        fprintf(stderr, "Não é possível abrir o arquivo de entrada de texto %s\n",argv[4]);
        exit(1);
    }
    int msg_escondida_length=get_mensagem_length(id_msg);


/*
Depois que o deslocamento foi lido e o cabeçalho do arquivo foi escrito como está para a imagem virgem - o comprimento da mensag
*/
    fputc(msg_escondida_length,id_msg_escondida);
    c++;
```

O “for” abaixo fica responsável por fazer uma varredura pela imagem e conseguindo assim obter cada bit de cada byte.

Com o uso da variável pedaco_msg, que captura cada caractere da mensagem que será passada e da variável arq_buffer que será comparada com o pedaco_msg.

Caso as variáveis sejam iguais, a função fputc(), função essa que escreve o caractere c (convertido para unsigned char) no fluxo de saída será chamada.

```c
171         fputc(msg_escondida_length,id_msg_escondida);
172         c++;
173         do {
174             int pedaco_msg;
175             if(!feof(id_msg)) {
176                 msg_buffer=fgetc(id_msg);
177                 for(int i=1;i<=8;i++) {  // Faça isso para cada bit em cada byte da imagem virgem
178
179                     arq_buffer=fgetc(id_arquivo);
180                     c++;
181                     int arq_byte = arq_buffer & 1; // E COM 1 PARA OBTER O VALOR DO BIT. E FAZ 0 SE O BIT FOR 0 OU 1 SE FOR
182
183                     pedaco_msg=get_bit(msg_buffer,i);
184                     //pedaco_msg=tabela_mascarada[i] & msg_buffer;
185                     if(arq_byte==pedaco_msg) {
186                         fputc(arq_buffer,id_msg_escondida);
187                     }
188                     else {
189                         if(arq_byte==0)
190                             arq_buffer = (arq_buffer | 1);
191                         else
192                             arq_buffer = (arq_buffer & ~1);
193                         // lógica para inverter o bit de arq_buffer e colocá-lo em um arquivo com putc ()
194                         fputc(arq_buffer,id_msg_escondida);
195                     }
196                 }
197             }
198             else {
199                 tmp_sig_cpy=fgetc(id_arquivo);
200                 fputc(tmp_sig_cpy,id_msg_escondida);
201                 c++;
202             }
203         } while(!feof(id_arquivo));
204         fclose(id_msg);
```

Neste segundo “for” temos todos os procedimentos para que possam ser juntados os bits e bytes da imagem passada anteriormente.

Após isso, os arquivos são fechados para que não fiquem como processo sendo utilizado na memória.

```
    else {
        id_msg=fopen(argv[4],"w");
        if (id_msg== NULL) {
            fprintf(stderr,"Não é possivel abrir o arquivo de entrada de texto%s\n",argv[4]);
            exit(1);
        }

        /* Grab BIT de todos os bytes para o comprimento especificado em fgetc */
        int msg_length=fgetc(id_arquivo);
        for(int i=0;i<msg_length;i++) {
            char temp_ch='\0';
            for( int j=0;j<8;j++) {
                temp_ch=temp_ch<<1;
                arq_buffer=fgetc(id_arquivo);
                int arq_byte = arq_buffer & 1;
                temp_ch|=arq_byte;
            }
            fputc(temp_ch,id_msg);
        }
        fclose(id_msg);
    }

    /* Limpe antes de sair */
    fclose(id_arquivo);
    fclose(id_msg_escondida);
    // return fuse_main(argc, argv, &fuse_example_operations, NULL);

}
```

Abaixo foram utilizadas as seguintes bibliotecas:

```
#define FUSE_USE_VERSION 26

#include <fuse.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
```

Ao tentar executar o código usando o FUSE, tivemos os seguintes erros:

```
ygor@ygor-VirtualBox:~/Documentos/SO/Projeto_SO$ cmake -DCMAKE_BUILD_TYPE=Debug .
-- The C compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
CMake Error at CMakeLists.txt:12 (find_package):
  By not providing "FindFUSE.cmake" in CMAKE_MODULE_PATH this project has
  asked CMake to find a package configuration file provided by "FUSE", but
  CMake did not find one.

  Could not find a package configuration file provided by "FUSE" with any of
  the following names:

    FUSEConfig.cmake
    fuse-config.cmake

  Add the installation prefix of "FUSE" to CMAKE_PREFIX_PATH or set
  "FUSE_DIR" to a directory containing one of the above files.  If "FUSE"
  provides a separate development package or SDK, be sure it has been
  installed.


-- Configuring incomplete, errors occurred!
See also "/home/ygor/Documentos/SO/Projeto_SO/CMakeFiles/CMakeOutput.log".
ygor@ygor-VirtualBox:~/Documentos/SO/Projeto_SO$ make -j
gcc -o main main.c
In file included from /usr/include/fuse/fuse.h:26,
                 from /usr/include/fuse.h:9,
                 from main.c:3:
/usr/include/fuse/fuse_common.h:33:2: error: #error Please add -D_FILE_OFFSET_BITS=64 to your compile flags!
   33 | #error Please add -D_FILE_OFFSET_BITS=64 to your compile flags!
      |  ^~~~~
main.c: In function ‘main’:
main.c:231:61: error: macro "fuse_main" passed 4 arguments, but takes just 3
  231 |  return fuse_main(argc, argv, &fuse_example_operations, NULL);
      |                                                             ^
In file included from /usr/include/fuse.h:9,
                 from main.c:3:
/usr/include/fuse/fuse.h:1015: note: macro "fuse_main" defined here
 1015 | #    define fuse_main(argc, argv, op)    \
      |
```

```
for( int j=0;j<8;j++) {
#define fuse_main(argc,argv,op,user_data) fuse_main_real(argc, argv, op, size
of(*(op)), user_data)

Main function of FUSE.This is for the lazy. This is all that has to be called from themain() function.This
function does the following:- parses command line options (-d -s and -h)- passes relevant mount options
to the fuse_mount()- installs signal handlers for INT, HUP, TERM and PIPE- registers an exit handler to
unmount the filesystem on program exit- creates a fuse handle- registers the operations- calls either
the single-threaded or the multi-threaded event loopNote: this is currently implemented as a macro.

Parâmetros:
 argc – the argument counter passed to the main() function
 argv – the argument vector passed to the main() function
 op – the file system operation
 user_data – user data supplied in the context during the init() method
}
fcl
}

/* Limp
fclose(
fclose(
return fuse_main(argc, argv, &fuse_example_operations);
```

Conclusão

Conseguimos atingir boa parte do que foi proposto, já que não foi possível executar os comandos necessários para a execução da parte do FUSE, porém, conseguimos implementar a esteganografia e entender de uma maneira melhor como é feita a esteganografia de arquivos.

Referências

Documentação do fuse - libfuse/libfuse: The reference implementation of the Linux FUSE (Filesystem in Userspace) interface (github.com)
URFJ - https://www.gta.ufrj.br/ensino/eel878/redes1-2016-1/16_1/esteganografia/
Wikipedia - https://en.wikipedia.org/wiki/Filesystem_in_Userspace
Engineering - https://engineering.facile.it/blog/eng/write-filesystem-fuse/
GitHub sobre leitura de arquivos BMP - Sadimanna - https://github.com/sadimanna/compression/blob/master/readbmp.c