# ATIVIDADE DE LABORATÓRIO - DECIFRA (BRUTE FORCE ATTACK)

Victor Dallagnol Bento
Universidade Federal de Santa Maria
Santa Maria - RS, Brasil
victor.bento@ecomp.ufsm.br

## I. INTRODUÇÃO

Neste laboratório, o professor nos incumbio com a tarefa de decifrar um mensagem em hexadecimal utilizando o código já desenvolvido, o RC4.

Nos foi dada a mensagem em hexadecimal, sendo que o último caractere da mensagem seria o ponto (.). Também foi informado que a chave continha quatro caracteres e que os mesmos estavam entre as letras do alfabeto (A/z a Z/z).

## II. DESENVOLVIMENTO TEÓRICO

O Ataque de Força Bruta consiste em tentar todas as possíveis senhas (também conhecidas como chaves) até se identificar a correta e, assim, conseguir acesso ao sistema. Qualquer equipamento que possua acesso à internet e necessite de usuário e senha pode ser alvo de um ataque de força bruta.

Em geral, esses ataques têm como objetivo identificar uma senha que está criptografada ou tentar acesso a um determinado sistema que necessita de autenticação, utilizando como base várias tentativas de senha consecutivas até que se acerte a verdadeira.

Embora esse ataque possa realizado manualmente, o que demanda uma quantidade alta de tempo, na maioria dos casos ele é realizado com o uso de ferramentas que permitem acelerar e automatizar a descoberta da senha, ocasionando uma futura invasão a um sistema.

## III. DESENVOLVIMENTO EXPERIMENTAL

Considerando os requisitos da tarefas citados na **Introdução** e utilizando o código RC4 deu-se início as modificações no código para decifrar a mensagem.

A função principal apresentada na **Figura 1** é a main, onde temos a entrada em hexadecimal, seu tamanho, e é feito um laço para que a entrada seja convertida para string, através de um cálculo utilizando a tabela ASCII.

Após a conversão, o novo tamanho da mensagem já convertida para string é salvo. Um arquivo é aberto para salvar os resultados e então é invocada a função para encontrar a chave.

```
//============================   Principal   ================================
void main(int argc, char const *argv[])
{
    // para loop "for"
    int i;

    // entrada em hexa
    char hex[] = "A3B1E7428957C942EC959C4F5DE56A7947F6CE18";
    int sizeHex = strlen(hex);

    // variaveis de conveniencia
    char ch, high, low;

    // trata todos os caracteres da entrada
    // Conversão de HEXA para STRING através da tabela ASCII
    for (i = 0; i < sizeHex; i += 2)
    {
       high = hex[i];
       high -= 0x30;
       if (high > 9) high -= 7;
       high <<= 4;

       low = hex[i+1];
       low -= 0x30;
       if (low > 9) low -= 7;

       ch = high | low;

       // transfere para saida
       input[i/2] = ch;
    }
    printf("<!> input in hexa: %s\n", hex);
    printf("<!> hexa to string: %s\n", input);
    len_input = strlen(input);  // Salva tamanho da mensagem

    fp = fopen ("Results.txt", "w");
    find_key();
    fclose(fp);

}
```

**Figura 1:** Função main.

A função *find_key* apresentada na **Figura 2** consiste em 4 laços de repetição, onde cada um deles representa um caractere da chave, no último laço são invocadas as funções para efetuar a decifra da mensagem.

```
//==============================   find_Key
void find_key()
{
    for (int a = 65; a <= 90; ++a)
    {
        key[0] = (char)a;
        for (int b = 65; b <= 90; ++b)
        {
            key[1] = (char)b;
            for (int c = 65; c <= 90; ++c)
            {
                key[2] = (char)c;
                for (int d = 65; d <= 90; ++d)
                {
                    key[3] = (char)d;
                    init();
                    stream();
                }
            }
        }
    }
}
```

**Figura 2:** Função *find_key*.

Todo processo de decifra continua normal, para cada combinação de chave gerada pelos laços de repetição. No final existe uma condição para apenas escrever no arquivo ,gerado anteriormente, se no final da mensagem tivermos o caractere ponto “.” e se os dois primeiros e os dois últimos caracteres forem letras do alfabeto (A/a até Z/z). O trecho da função pode ser visto na **Figura 3**.

```
if (result[len_input - 1] == '.')
{
    if ((result[0] >= 65 && result[0] <= 90) || result[0] >= 97 && result[0] <= 122)
        if ((result[1] >= 65 && result[1] <= 90) || result[1] >= 97 && result[1] <= 122)
            if ((result[len_input-2] >= 65 && result[len_input-2] <= 90) || result[len_in
                if ((result[len_input-3] >= 65 && result[len_input-3] <= 90) || result[le
                    fprintf(fp, "->\t%s\tChave: %s\n", result, key);
}
```

**Figura 3:** Condições para escrita dos resultados no arquivo.

No final da função, acontece o retorno para a função principal (main) onde o arquivo com os resultados é fechado e o programa encerrado.

## IV. CONCLUSÕES

Ao final do programa, abrindo o arquivo gerado pelo mesmo (*Results.txt*), podemos verificar a mensagem decifrada, assim como a chave que foi usada para decifra-la. A **Figura 4** mostra os resultados.

```
->  ik·EŠŠ[00][19]íÒ: Chave: AHHU[00][14]
->  Parabéns, conseguiu.    Chave: DAMZ[00][14]
->  Mdôm[00][8E]÷bÆûÈ[00][12][00][16]KŽ[00][11]?\tT.    Chave: FWCM[00][14]
->  wCHóÐ[00][9E]×-[00][80]yE[00][8E]Ù[00][1A]ËB FC.    Chave: TZTU[00][14]
->  CkNEVWû<#u*Mcí¶[00][18][00][85]rj.  Chave: UEEV[00][14]
```

**Figura 4:** Resultados obtidos.

## REFERÊNCIAS


[1] **Ataque de Força Bruta (Brute Force Attack)**, https://canalcienciascriminais.com.br/ataque-de-forca-bruta/ acessado em 17.04.2019.

[2] Material disponibilizado em aula.