Quebra de senhas

Henrique Hepp
Universidade Federal do Paraná
hhepp@inf.ufpr.br

ABSTRACT

Esse trabalho descreve como duas senhas criptografadas de dois usuários foram descriptografadas. Como entrada do problema, foi dado um hash em Md5, uma cadeia de caracteres e um arquivo com os usuários e as senhas. Por força bruta descobriu-se a palavra chave com a qual foi feito o MD5 dado; com essa chave foi decodificado a cadeia de caracteres; a cadeia decodificada deu uma dica que foi usada para criar um dicionário de palavras, que por sua vez foi usado com o programa John the Ripper para decodificar o arquivo com os usuários e senhas.

Keywords

Senha, criptografia, John the Ripper, dicionário, C

1. INTRODUÇÃO

Esse trabalho descreve como duas senhas criptografadas de dois usuários foram descriptografadas. Foram disponibilizados: um hash em Md5, uma cadeia de números hexadecimais e um arquivo com os usuários e as senhas cifradas. O "modus operandi" indicado foi: primeiro, descobrir por força bruta uma chave com a qual foi feito o MD5. Depois decodificar a cadeia de números hexadecimais com essa chave usando a operação xor. Para fazer essa etapa escreveu-se um programa em c nomeado de brutexor.c. Essa decodificação oferece uma dica para poder realizar um download de vários arquivos textos. Esses arquivos textos foram usados para criar um dicionário de palavras. Para essa etapa escreveu-se um program em c nomeado de wordlist.c. Por último usou-se esse dicionário com o programa John the Ripper para decodificar as senhas do arquivo oferecido.

Nas próximas seções serão vistos os códigos implementados, os testes feitos e os resultados obtidos.

2. IMPLEMENTAÇÃO

Para as duas primeiras partes, a geração de listas de palavras e obtendo dicas com XOR, foram escritos dois programas na linguagem c descritos em seguida.

2.1 Geração de Listas de Palavras

Foi escrito um programa em c que recebe como argumento o nome de um diretório, lê todos os arquivos nesse diretório e cria um arquivo "dicionário.txt" com todas as palavras sem repetição de todos os arquivos. Definiu-se que esses arquivos devem estar na codificação UTF-8. Para converter os arquivos pode-se usar o programa "iconv" com um comando no terminal. Por exemplo, para arquivos em ISO-8859-1, usou-se:

```
for file in *.txt; do
    iconv -f ISO-8859-1 -t UTF-8 -o "$file"
        .utf "$file" && mv "$file".utf "
        $file"
done
```

Dentro do programa em c, para usar a codificação UTF8 usou-se as bibliotecas "wchar.h" e "locale.h" e setou-se o local com:

```
setlocale(LC_ALL, "pt_BR.UTF-8");
```

Para abrir o diretório e iterar sobre os arquivos nesse diretório usou-se a biblioteca "dirent.h" e os comandos:

```
D = opendir(nomeDir))

while (file = readdir(D))
{
    sprintf(endeArquivo,"%s/%s",dir,file->
        d_name);
    entrada = fopen(endeArquivo, "r");
    //leitura dos arquivos e escrita do
        dicionario
      ...
}
```

Antes de ler os arquivos abriu-se o arquivo dicionário com o comando:

```
//abre o dicionario
saida = fopen("dicionario.txt","a+");
```

Agora, cada arquivo texto é separado em palavras. Captase as letras com a função "fgetwc" e enquanto for uma letra da tabela UTF8, ou um hífen, compõe-se a palavra. Quando se encontra um espaço ou pontuação, segue-se adiante com a escrita do dicionário. Isso é repetido até terminar o arquivo texto.

ACM ISBN xxx-xxxx-xx-xxx.
DOI: xx.xxxx/xxxxx

```
do{
   //separa em palavras
   while ( (wc = fgetwc(entrada)) != WEOF
       && ( (wc >= 'A' && wc <= 'Z') || (wc
       >= 'a' && wc <= 'z') || (wc >= 192)
        || (wc=='-') ) )
      palavra[i]=wc;

   //procura no dicionario
      ...
   //escreve no dicionario
      ...
//ate o final do arquivo
}while(wc != WEOF);
```

Cada palavra lida é procurada no dicionario.txt, se ela não for encontrada é então adicionada.

```
rewind(saida);//comeca a ler no inicio do
    arquivo
while ( fgetws ( line, TAM , saida) != NULL
    )
     if (wcscmp(palavra,line) == 0 )
          encontrado = TRUE;

//se nao achar a palavra escreve em
    dicionario
if (encontrado==FALSE)
     fwprintf (saida, L"%ls\n",palavra);
```

2.2 Obtendo dicas com XOR

Nessa parte do trabalho, o programa em c, nomeado por brutexor.c, recebe como entrada do stdin um arquivo texto em que a primeira é um hash de MD5 e a segunda linha é uma cadeia de números hexadecimais. Por força bruta, procura-se descobrir a chave com a qual o hash foi feito. Depois com essa chave decifra-se a cadeia de números.

A leitura do hash e da cadeia é feita com o código abaixo:

```
for(i = 0; i < MD5_DIGEST_LENGTH; i++)
        scanf("%2x",&mdtest[i]);

scanf("%2x",&cadeia[0]);
i=1;
while(scanf("␣%2x",&cadeia[i++])!=EOF);
int tamCadeia=i-1;
```

Para encontrar a chave com a qual o hash foi feito, cria-se todas as sequências possíveis com letras maiúsculas e minúsculas de tamanho 1 até nMax, que nesse caso é 8. Com cada uma dessas sequências é feito o md5 e esse comparado com o lido. Como mostra o for abaixo:

```
for (n=1;n<=nMax;n++){
//cria todas as sequencias de 1 ate n
        ...
//para cada sequencia faz o md5 e compara
    com o lido
        ...
//se achou a chave decifra a cadeia
        ...
}
```

Para criar/enumerar todas as sequências de tamanho n usa-se o seguinte algoritmo: Primeiro, cria-se uma sequencia numérica com n posições e m possíveis dígitos, de 0 até m-1. Então, encontra-se a próxima sequencia somando 1, repete-se até encontrar todas sequencias. Por exemplo, se n=3 e m=2, começamos com 000; depois para cada iteração somamos 1; as próximas sequencias seriam então 001, 010, ..., 111.

Para usar a sequencia numérica, precisa-se converter ela em uma sequencia de letras. Nesse caso, cria-se um vetor (nomeado por objetos[]) com todas as letras maiúsculas e minúsculas. Cada número da sequência numérica é substituído por uma letra desse vetor cujo índice seja igual a esse número. Por exemplo, a sequência 001 seria transformada para objetos[0] objetos[0] objetos[1], que seria, AAB.

O código desse algoritmo está abaixo:

```
//cria a primeira sequencia
for (i=0; i<n; i++)
        seq[i] = 0;

do {
//converte a sequencia numerica na
    sequencia de letras
  for (i=0; i<n;i++){
        password[i]=objetos[seq[i]];
  }password[i]='\0';

  //faz o md5 de password e compara
  ...
  //se achou decifra a cadeia
  ...

  //gera a proxima sequencia
} while (proxima(seq, n, m));
```

Para encontrar a próxima sequencia soma-se um.

```
int proxima(int seq[], int n, int m) {
    int i = n-1;
    while (i>= 0) {
        seq[i] = (seq[i] +1) % m;
        if (seq[i] == 0)
                i--;
        else
                return TRUE;
     }
     return FALSE;
}
```

Para cada uma das sequencias de letras, chamada agora de password, calcula-se o md5. Para tal usa-se uma função da biblioteca "openssl/md5.h".

```
MD5((unsigned char *)password, strlen(
    password), md);
```

Esse hash calculado é comparado com o recebido pelo programa. Se ele for igual faz-se um xor da chave com os caracteres da cadeia. O resultado é impresso e com isso o programa é finalizado.

```
for (i=0; i<tamCadeia; i++)
 printf("%c",password[i%strlen(password)]^
     cadeia[i]);
```

2.3 Força-bruta por Dicionário

Com o dicionario recebido pela parte "Geração de Listas de Palavras", foi executado o programa John The Ripper no terminal com o comando:

```
john --wordlist=./dicionario.txt ./
    _etc_shadow
```

3. TESTES E RESULTADOS

3.1 Obtendo dicas com XOR

Tendo como entrada o hash 8d7b356eae43adcd6ad3ee124c3dcf1e e a cadeia 0b 34 30 20 2f 27 05 22 05 29 21 28 22 26 19 34 23 22 27 2d, obteve-se como saída do programa a palavra chave “FACIL” e a dica MusicaDaLegiaoUrbana, como se pode ver pela figura 1.

3.2 Geração de Listas de Palavras

Usando a dica “MusicaDaLegiaoUrbana” encontrada pelo passo anterior, fez-se o download de todas as letras da legião urbana disponíveis no site cifras.com.br. Essas letras, com as cifras, foram transformadas da codificação ISO-8859-1 para a UTF8. Então, executando o programa wordlist.c, obteve-se o dicionário das palavras dessas letras, totalizando 4525 palavras.

3.3 Força-bruta por Dicionário

Usando o dicionário montado no passo anterior executou-se o programa John the Ripper para decodificar o arquivo etcShadow. Foram encontradas as senhas base para o usuário base e a senha Cavalos-marinhos para o usuário eva. Não foi encontrada a senha do usuário beto. Abaixo temos a saída do programa.

```
john --wordlist=./dicionario.txt ./
    etcShadow
Loaded 3 password hashes with 3 different
    salts (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any
    other key for status
base                (base)
Cavalos-marinhos (eva)
2g 0:00:00:33 100\% 0.05913g/s 133.7p/s
    346.6c/s 346.6C/s percebo..transcrita
```

4. CONCLUSÃO

Esse trabalho descreveu como duas senhas criptografadas de dois usuários foram descriptografadas. Foram oferecidas, um hash em Md5, uma cadeia de caracteres e um arquivo com os usuários e as senhas. Fez-se um programa em para por força bruta descobrir a palavra chave com a qual foi feito o MD5 dado. Com a chave descoberta foi decodificado a cadeia de caracteres usando a operação xor. A cadeia decodificada deu uma dica que foi usada para criar um dicionário de palavras, que por sua vez foi usado com o programa John the Ripper para decodificar com sucesso duas senhas, uma, no entanto, não foi descoberta.

Como trabalhos futuros, sugere-se a implementação de um programa que crie dicionários com palavras em outras codificações além de UTF8. Também precisaria-se descobrir como decodificar a terceira senha não descoberta.

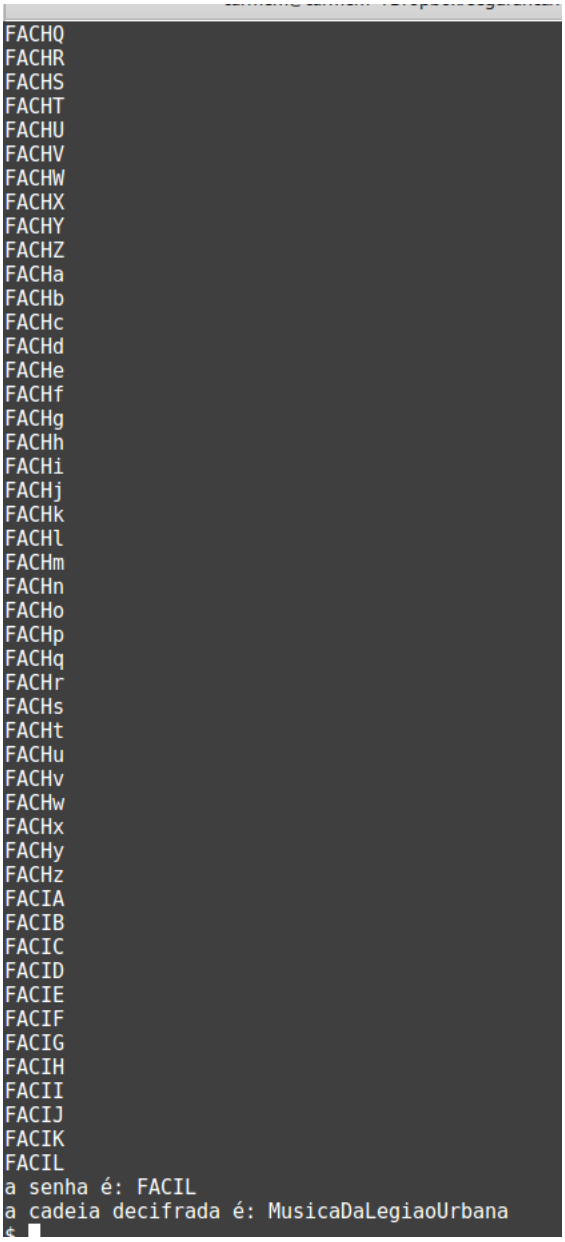

Figure 1: Saída do programa brutexor.c

5. REFERENCES

[1] M. M. Sanches. *Algoritmos de enumeração*, 2012.