Esteganografia

Aprenda como esconder uma mensagem em plena vista e suas aplicações em CTFs

- Cifra de Bacon
- Interpretar imagem como texto
- LSB: Least Significant Bits
- Imagem em áudio
- O que é esteganografia?

Cifra de Bacon

Nessa seção veremos um primeiro exemplo de Esteganografia: a **Cifra de Bacon**.

> “ Ela tem esse nome pois foi criada por Francis Bacon em 1605.

A ideia por trás dessa cifra, diferente de uma cifra criptográfica comum, é esconder a mensagem secreta por meio de um texto legível, apenas mudando a grafia de suas letras.

Essa é uma cifra de substituição, e cada letra é representada por um conjunto de 5 caracteres binários ('a' e 'b' ou '0' e '1'). O **alfabeto de substituição** da Cifra de Bacon possui dois modelos:

- **A cifra de 24 letras:** É a original. Nela, os pares de caracteres (I,J) e (U,V) não possuem distinção.

```
A = aaaaa  I/J = abaaa    R = baaaa
B = aaaab    K = abaab    S = baaab
C = aaaba    L = ababa    T = baaba
D = aaabb    M = ababb  U/V = baabb
E = aabaa    N = abbaa    W = babaa
F = aabab    O = abbab    X = babab
G = aabba    P = abbba    Y = babba
H = aabbb    Q = abbbb    Z = babbb
```

- **A cifra de 26 letras:** A segunda versão da cifra. Agora todas as letras possuem um código único.

```
A = aaaaa  I = abaaa  Q = baaaa  Y = bbaaa
B = aaaab  J = abaab  R = baaab  Z = bbaab
C = aaaba  K = ababa  S = baaba
D = aaabb  L = ababb  T = baabb
E = aabaa  M = abbaa  U = babaa
F = aabab  N = abbab  V = babab
G = aabba  O = abbba  W = babba
```

```
H = aabbb  P = abbbb  X = babbb
```

Dessa forma, substituímos o 'A' da mensagem secreta por 'aaaaa', um 'B' por 'aaaab' e assim por diante. Por exemplo, digamos que queremos esconder a mensagem `fly you fools`. Assim, a substituição das letras será da forma abaixo, usando a cifra de 26 letras:

```
texto original: F     L     Y     Y     O     U     F     O     O     L     S
texto cifrado:  aabab ababb bbaaa bbaaa abbba babaa aabab abbba abbba ababb baaba
```

Com isso, podemos usar esse padrão e esconder em uma mensagem comum de tamanho maior ou igual ao texto cifrado, como `according to all known laws of aviation, there is no way a bee should be able to fly`.

Assim, removendo os espaços e pontuações para facilitar a associação, podemos esconder a mensagem no texto de várias formas, como associar letras **maiúsculas** ao 'a' e **minúsculas** ao 'b', associar a letras com ou sem **itálico** ou até com duas **fontes diferentes**. Para esse exemplo, usaremos maíusculas e minúsculas.

```
falsa mensagem: accordingtoallknownlawsofaviationthereisnowayabeeshouldbeabletofly
texto cifrado:  aababababbbbaaabbaaaabbbababaaaababababbbaabbbaababbbaaba
mensagem final: acCoRdInGTOAllkNOwnlaWSOfAvIatioNtHeREIsnOWAyaBeESHouLdbeabletofly
```

Voltando à formatação original da mensagem, temos: `acCoRdInG TO All kNOwn laWS Of AvIatioN, tHeRE Is nO WAy a BeE SHouLd be able to fly`. Com isso, quem interceptar essa mensagem esteganográfica, não vai imaginar que ela possui algo além de uma escrita engraçada.

Identificando

Em uma primeira observação, uma mensagem oculta pela cifra de Bacon pode ser identificada pela mudança alternada dos padrões das letras: maíuscula e minúscula, itálico, fonte, ou até a alternância explícita de duas letras.

Outra abordagem, seria uma análise de frequência. Como a cifra de Bacon é um tipo de cifra de substituição, ela é sensível a uma análise de frequência.

Solucionando

Para solucionar desafios que contém Cifra de Bacon, o primeiro passo é identificar o modo como a mensagem foi escondida. Depois disso, é preciso percorrer o texto e transformá-lo em sequências de 'a's e 'b's.

Assim, você pode guardar o alfabeto de substituição com um dicionário e usá-lo para substituir os blocos de 5 caracteres pela letra correspondente.

Por exemplo, podemos implementar isso com um código em Python:

```
bacon_to_letter_26 = {
    'aaaaa':'A', 'aaaab':'B', 'aaaba':'C', 'aaabb':'D', 'aabaa':'E',
    'aabab':'F', 'aabba':'G', 'aabbb':'H', 'abaaa':'I', 'abaab':'J',
    'ababa':'K', 'ababb':'L', 'abbaa':'M', 'abbab':'N', 'abbba':'O',
    'abbbb':'P', 'baaaa':'Q', 'baaab':'R', 'baaba':'S', 'baabb':'T',
    'babaa':'U', 'babab':'V', 'babba':'W', 'babbb':'X', 'bbaaa':'Y',
    'bbaab':'Z'
}

def format(text, a='a', b='b'):
    """Format a steganographic text to a binary sequence"""
    formated_text = ''
    for c in text:
        if not c.isalpha():
            continue
        if c.istitle():
            formated_text += b
        else:
            formated_text += a
    return formated_text

def decode(text, a='a', b='b', bacon_alpha=bacon_to_letter_26):
    """Decode a encrypted Bacon cipher text"""
    cipher = format(text, a=a, b=b)
    output = ''
```

```
    while len(cipher) >= 5:
        token, cipher = cipher[:5], cipher[5:]
        if token in bacon_alpha:
            output += bacon_alpha[token]
        else:
            break
    return output

if __name__ == '__main__':
    input  = input()
    output = decode(input)
    print(output)
```

Exercícios

WeChall: Baconian

WeChall: Bacon Returns

Referências

- GeeksforGeeks
- Practical Cryptography

Interpretar imagem como texto

Como foi explicado na seção de dados e códigos, qualquer arquivo no computador pode ser interpretado como uma sequência binária, e dessa forma, tem sua representação em texto. Assim, esse recurso pode ser usado para esconder informação numa imagem.

Um recurso comum em CTFs é colocar um trecho de texto puro em alguma região arbitrária da imagem. Por exemplo, se queremos esconder a palavra `such cake` na imagem `doge.jpg`, podemos usar o comando Linux:

```
echo "such cake" >> doge.jpg
```

Com isso, `"such cake"` ficará no final dos dados da imagem.

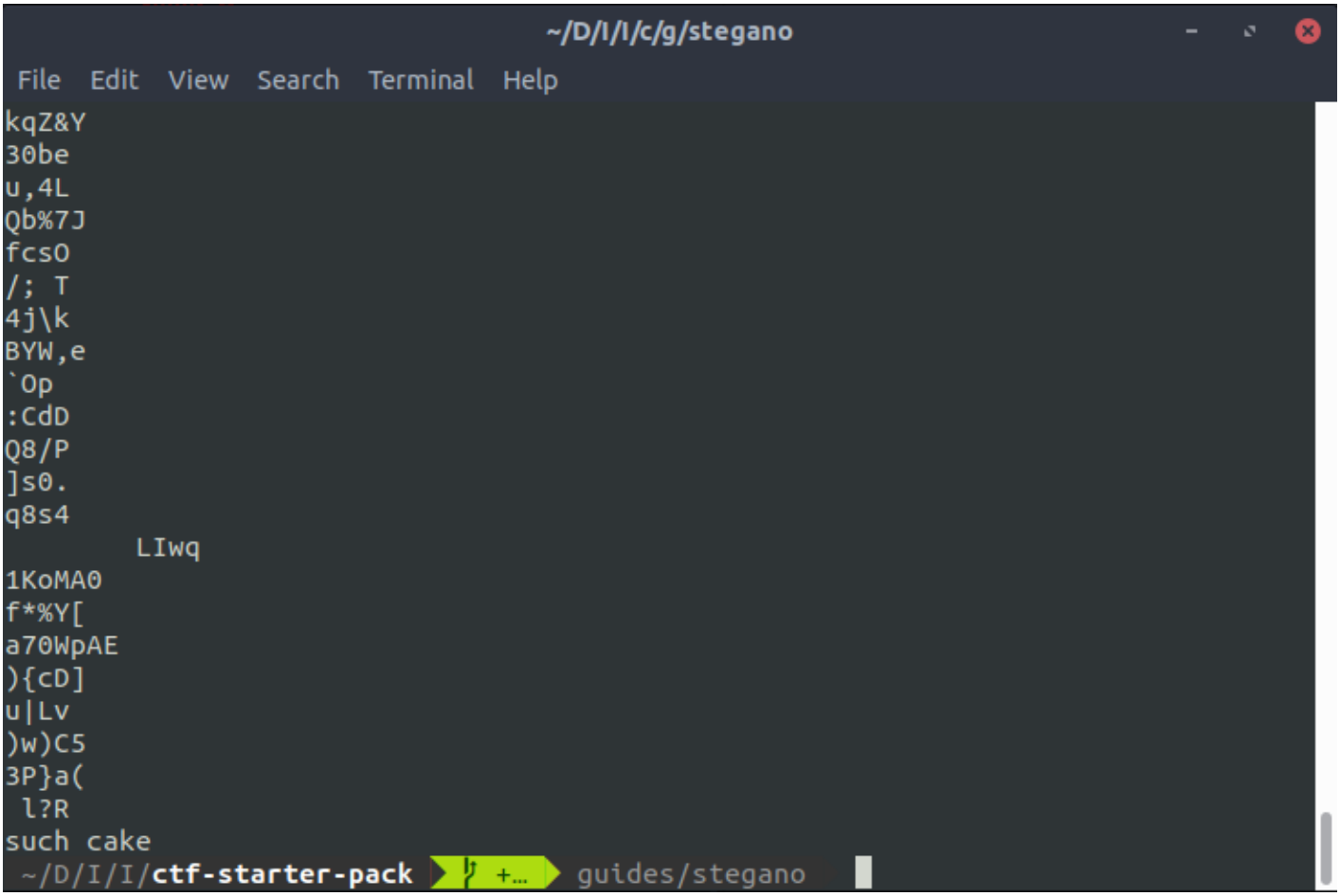
Solucionando

Para extrair a informação escondida por esse método, várias ferramentas podem ser usadas. As duas principais são os comandos de Linux `strings` e `hexdump`.

O comando `strings` imprime basicamente todas as sequências de dados *imprimíveis* de um arquivo. Por exemplo, o comando aplicado a imagem `doge.jpg`

```
strings doge.jpg
```

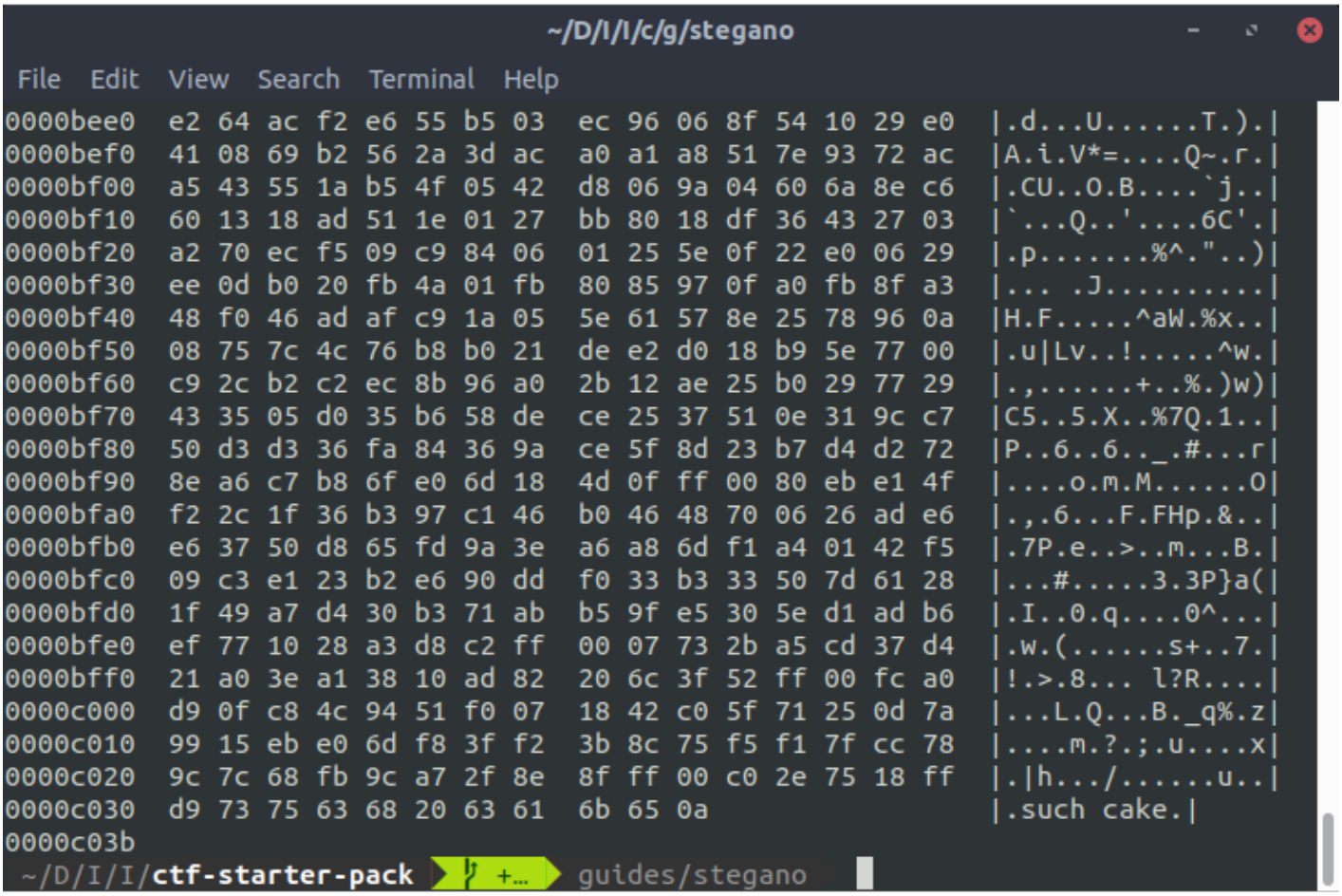
Com isso, o final da saída do comando será algo da forma:



Já o comando `hexdump` permite que a imagem seja analisada de forma mais minuciosa, onde o formato de leitura e impressão pode ser especificado pelo usuário. Por exemplo, um uso do comando com a imagem `doge.jpg`, onde a flag `-C` representa a forma canônica de impressão hex+ASCII:

```
hexdump -C doge.jpg
```

O final da saída desse comando é algo da forma:



Exercícios

WeChall: Stegano I

picoCTF-2018: hex-editor

Referências

HowtoForge

Sanfoudry

LSB: Least Significant Bits

Um dos métodos mais comuns e populares nos dias de hoje para se esconder uma mensagem numa imagem é a técnica de **LSB** ou **Least Significant Bits**. Esse método consiste em esconder a informação nos bits menos significativos de uma imagem.

Representação da imagem

Como mencionado em Dados e Códigos, todos os arquivos no computador são sequências binárias e com as imagens não é diferente.

Uma imagem é composta, na maioria das vezes, por duas partes. O **cabeçalho** (ou *header*) é onde ficam armazenadas as informações sobre a imagem, como o seu formato e dimensões. Já o **corpo** da imagem é onde ficam armazenados a informação dos seus *pixels* de fato.

Cada formato tem suas particularidades para representar um pixel, por simplicidade vamos considerar um pixel de 24 bits de uma imagem lossless.

Por exemplo, um pixel com a cor em RGB `#ebc700` teria sua representação em binário como `11101011 11000111 00000000` onde

Red	Green	Blue
8 bits	8 bits	8 bits
11101011	11000111	00000000

Escondendo na imagem por LSB

Para esconder informação em uma imagem usando o método de LSB, alteraremos os bits menos significativos de cada byte no corpo da imagem.

Por exemplo, se temos uma imagem de três pixels

```
pixel 1: 11101011 11000111 00000000
pixel 2: 01000010 10000110 11110100
pixel 3: 11110100 01010111 01000010
```

Se queremos esconder o byte `01100001`, uma nova imagem será gerada com a mensagem nos últimos bits (destacado por * *).

```
pixel 1: 1110101*0* 1100011*1* 0000000*1*
pixel 2: 0100001*0* 1000011*0* 1111010*0*
pixel 3: 1111010*0* 0101011*1* 01000010
```

Dessa forma, ao esconder a mensagem nos bits menos significativos, fazemos apenas uma pequena alteração na imagem. Por exexmplo, o `pixel 1` foi da cor `#ebc700` para `#eac701`.

Assim, vemos que a imagem precisa ser maior que a mensagem que vamos esconder nela. Nessa caso, no mínimo **8 vezes maior**. Para contornar esse problema, alguns algoritmos usam 2 ou 3 bits menos significativos de cada byte, aumentando a alteração da imagem.

Para recuperar a mensagem, fazemos o processo inverso: identificando os bits menos significativos no corpo e remontando em bytes.

```
pixel 1: 1110101*0* 1100011*1* 0000000*1*
pixel 2: 0100001*0* 1000011*0* 1111010*0*
pixel 3: 1111010*0* 0101011*1* 01000010

-> mensagem: 01100001
```

Ferramentas para LSB

Stéganô

O pacote Stéganô possui várias ferramentas para esteganografia. A maioria sua a técnica de LSB.

Ele necessita de **Python 3** e pode ser instalado por

```
sudo pip3 install Stegano
```

Uma das ferramentas desse pacote é o comando `stegano-lsb`.

Ele pode ser usado para esconder uma mensagem em uma imagem com o argumento `hide`, como por exemplo

```
stegano-lsb hide -i dog.png -o dog_steg.png -m "dont-worry-be-happy"
```

Já com o comando `reveal`, podemos recuperar a mensagem a partir da imagem.

```
stegano-lsb reveal -i dog_steg.png
```

Além disso, em vez de modificar os pixels sequncialmente a partir do começo, um outro comando, o `stegano-lsb-set`, permite inserir a informação de acordo com outros padrões especificados. Ele recebe um argumento para um gerador, como por exemplo `eratosthenes`, que produz os pontos onde os pixels serão modificados.

Assim, é preciso usar o mesmo gerador para esconder e recuperar a mensagem:

```
stegano-lsb-set hide -i dog.png -o dog_steg2.png -g eratosthenes -m "dont-worry-be-happy"

stegano-lsb-set reveal -i dog_steg2.png -g eratosthenes
```

zsteg

O zsteg é uma ótima e super simples ferramenta para **detectar** steganografia em imagem. Ela necessita de Ruby e pode ser instalada por

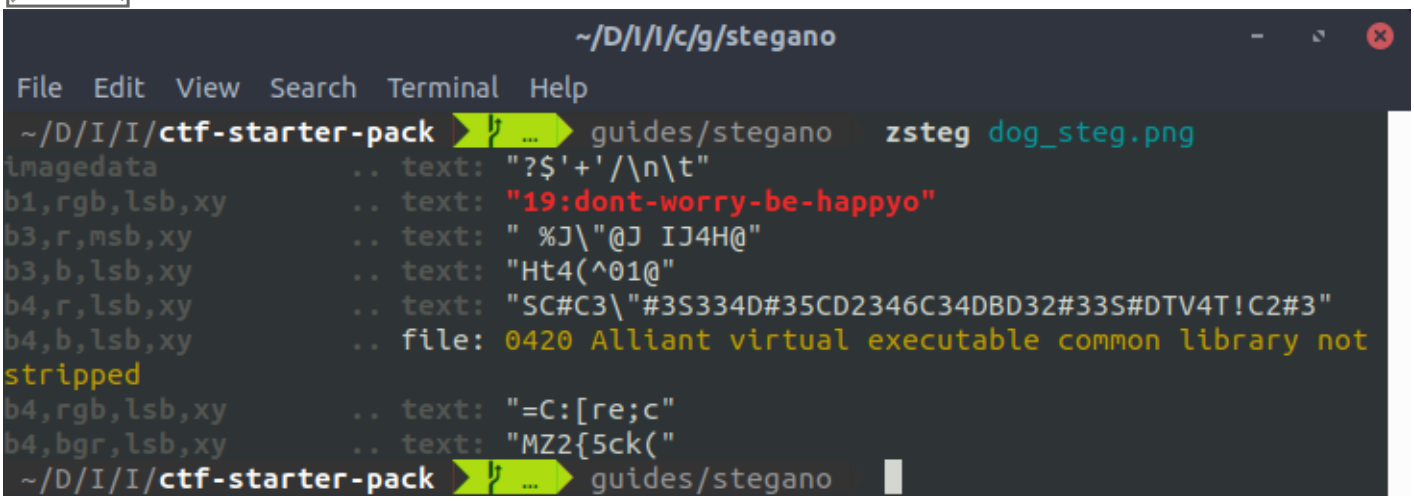
```
gem install zsteg
```

Para detectar uma informação escondida por um LSB simples, pode-se digitar apenas

```
zsteg dog_steg.png
```

O resultado será algo como

Image not found or type unknown



Informações adicionais

Muitas vezes, a técnica de LSB pode ser usada junto a uma **criptografia** para proteger ainda mais a mensagem e dificultar sua detecção.

Na vida real, a técnica de LSB pode ser usada como **marca d'água digital** para detectar cópias de mídias não autorizadas. O programa OpenStego é um ótimo exemplo de software com essa finalidade.

Exercícios

(to-do)

Referências

Computerphile

Stéganô

zsteg

Imagem em áudio

Mais uma forma interessante de esteganografia é esconder uma imagem em um áudio. Nela, a imagem é transformada em ondas sonoras de forma que ela pode ser vista através de seu **espectrograma**.

O espectrograma

Um espectrograma é uma forma de visualizar a intensidade de um sinal através do tempo e em várias frequências.

Os espectrogramas são gráficos de duas dimensões (frequência x tempo), com a terceira dimensão, a amplitude, sendo representada pela variação das cores. O **tempo** é normalmente representado no eixo x em sentido crescente. Já a **frequência** (medida em Hz) é representada no eixo y, com frequências mais baixas em baixo e mais altas em cima. Por fim, a **amplitude** (medida em dB) é representada com cores mais escuras para amplitudes menores e cores mais claras para maiores.

Para visualizar o espectrograma de um áudio, pode-se usar o programa sonic visualizer (mais simples) ou audacity (mais robusto).

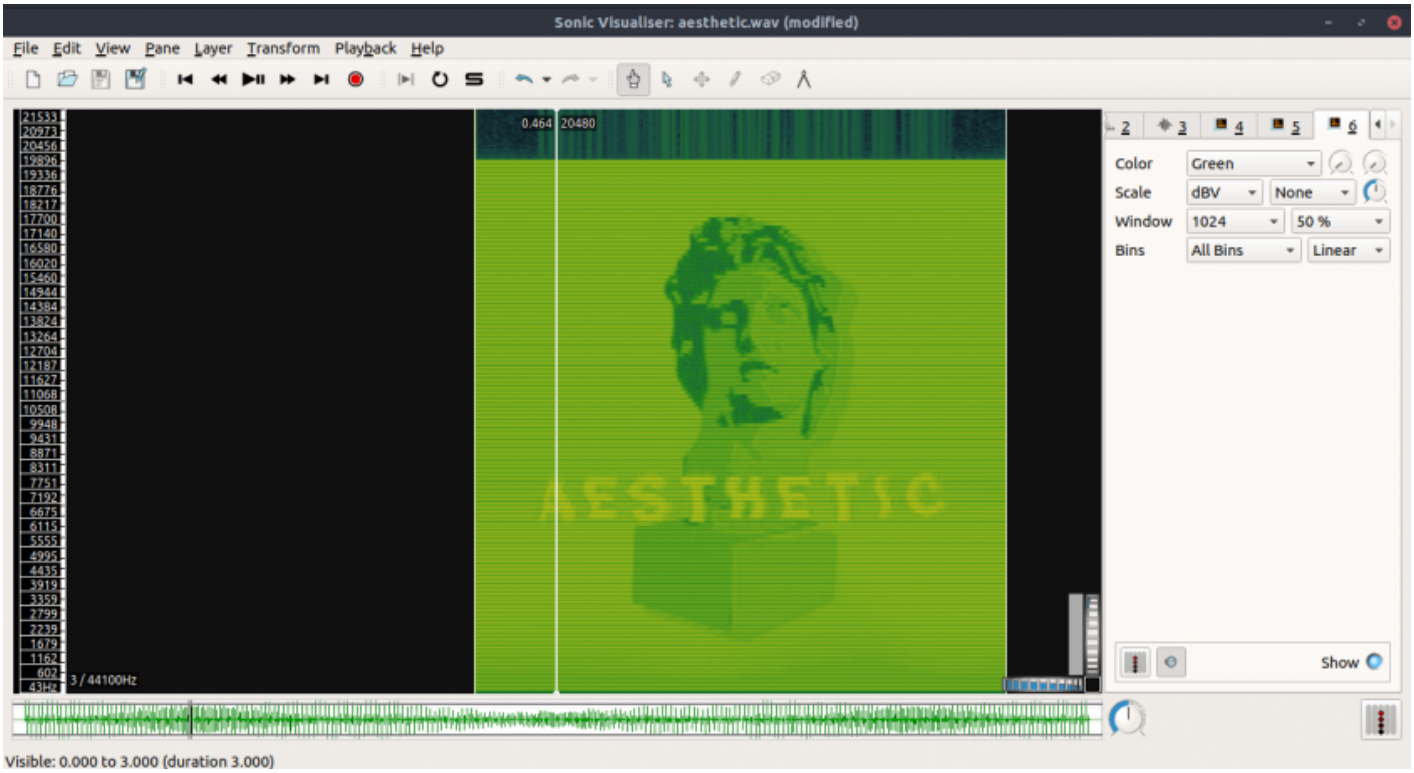
Escondendo uma imagem em um áudio

Primeiro, precisamos realizar a conversão da imagem para o áudio. Isso pode ser feito utilizando a ferramenta img-encoder, no GitHub.

Por exemplo, se queremos converter a imagem `aesthetic.png`

No demo online dessa ferramenta, adicionamos a imagem em `Open-Image...`, e convertemos com `Encode`. Renomearemos o resultado para `aesthetic.wav`.

Utilizando o `sonic-visualizer` para visualizar o áudio gerado, mudamos a visualização para `Add spectrogram`, com o comando `Shift+G`.

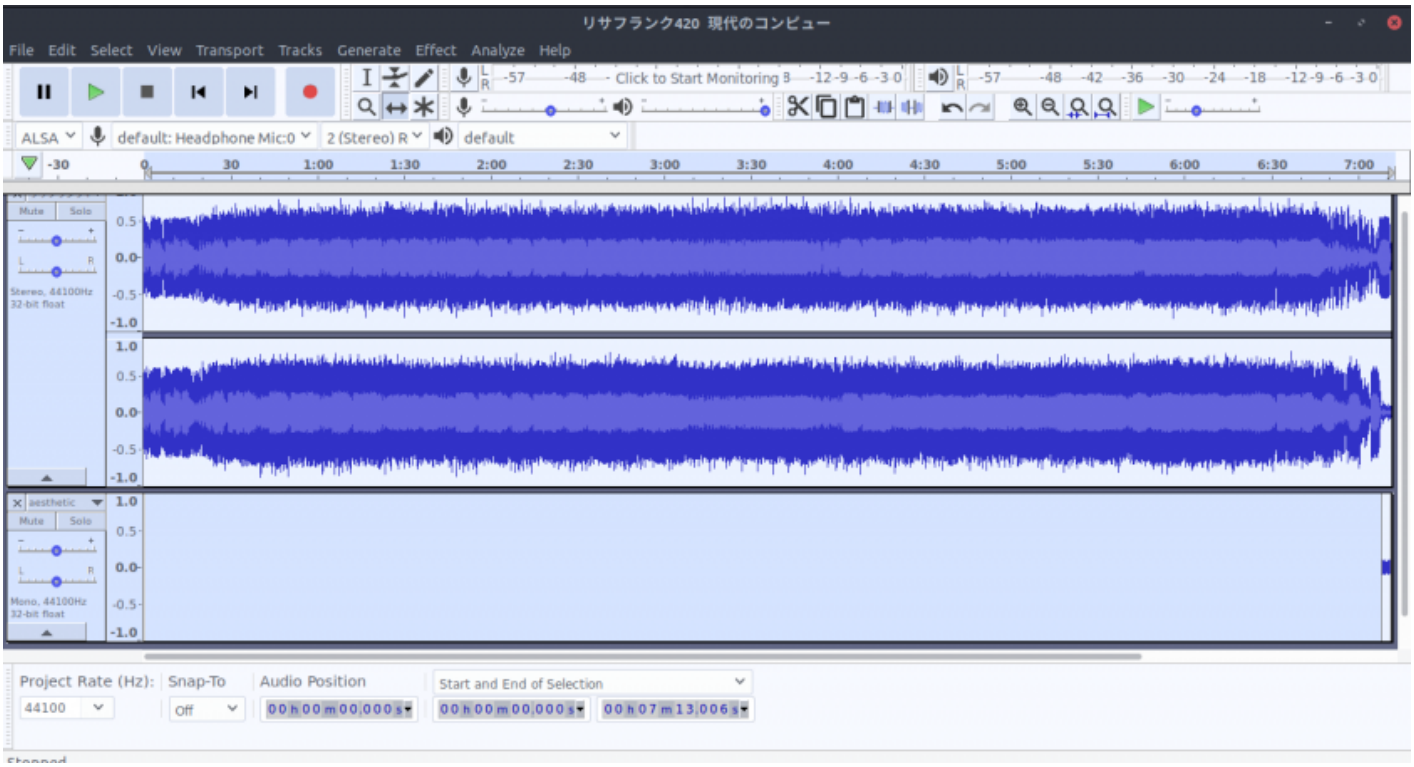


Agora, precisamos juntar esse áudio em outra música para camuflar a informação. Por exemplo, usaremos a música `420`. Abrindo a música com `audacity`, importamos o nosso áudio com `Ctrl+Shift+i`.

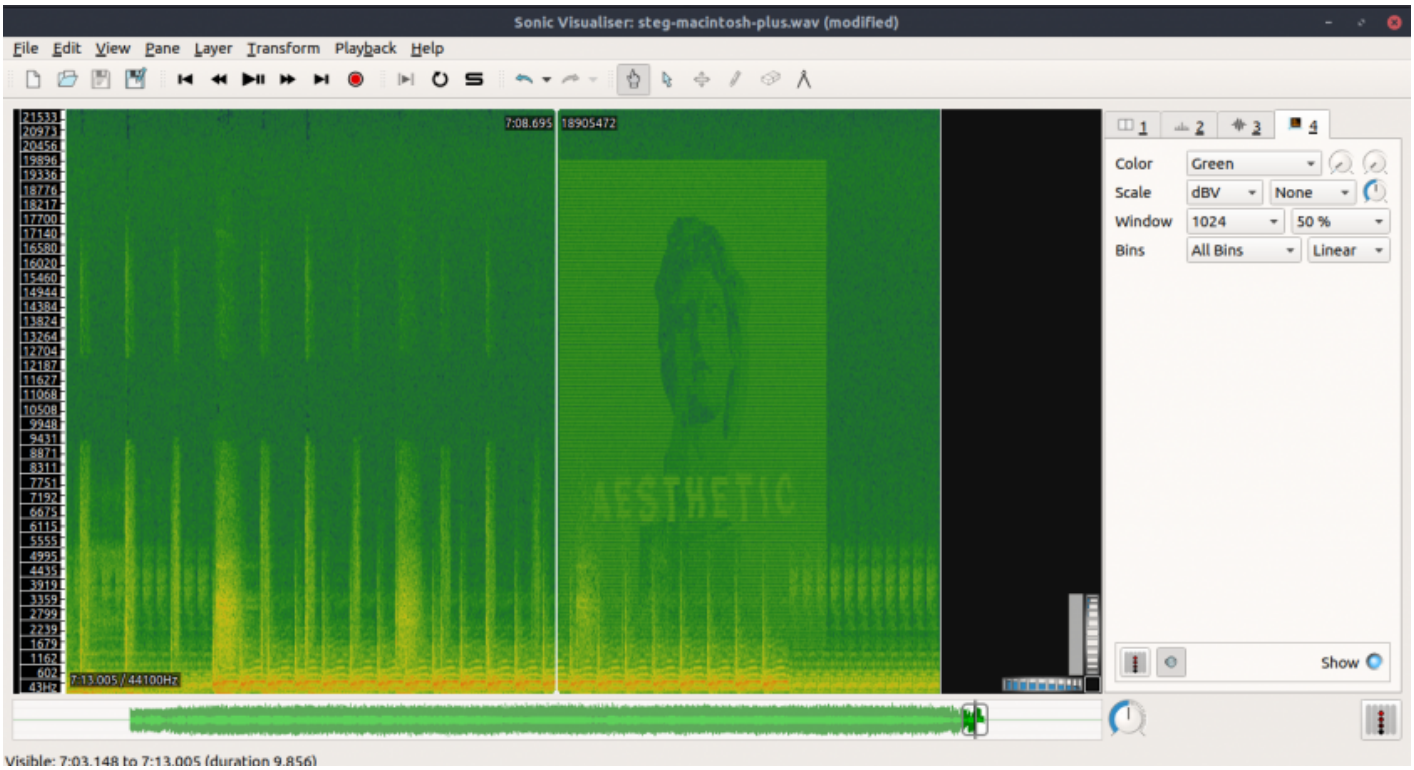
Com a ferramenta `Time Shift Tool`, podemos mudar a posição do áudio esteganográfico para o

começo ou fim (assim a música vai interferir menos o áudio).

Porém, o áudio esteganográfico ainda pode ser percebido devido a seu ruído. Para diminuir esse efeito, primeiro selecionamos a *track* do áudio (clicando em uma 'área neutra' do Painel de Controle, na parte esquerda da *track*) e clicando em `Effect` e `Amplify`. Assim, podemos mudar a amplitude do som para dificultar sua detecção (algo em torno de -20 dB).



Assim, selecinando todas as *tracks* e exportando a música gerada para WAV, `steg-macintosh-plus.wav`, teremos uma música audível mas com uma imagem escondida em seu espectrograma.



Referências

PNSN: Espectrogramas

Selecionando Tracks no Audacity

Mental Floss

O que é esteganografia?

Em um primeiro contato com a área, **esteganografia** provavelmente será o nome que lhe causará mais estranheza. O termo vem do grego *steganos*, que significa "encoberto, escondido" e *graphein*, que significa "escrita".

Esteganografia é a técnica de ocultar uma mensagem dentro de outra. Diferentemente da criptografia, onde torna-se a mensagem ilegível, a esteganografia procura **esconder o fato de que a mensagem existe** em primeiro lugar.

Assim, a falsa mensagem funcionará como uma espécie de cobertura da mensagem real, desviando a atenção de quem estiver buscando a mensagem real. Alguns exemplos poderiam ser o ato de escrever com tinta invisível entre as linhas de uma carta comum ou escrever um texto nos bits de uma imagem sem alterá-la visualmente.

Nas próximas seções, serão abordadas técnicas de esteganografia e suas aplicações em CTFs.