Logotipo

Descrição gerada automaticamente com confiança baixa

Relatório de CTF

Breaking Crypto the Simple Way – TryHackMe

|  |  |
| --- | --- |
| **Informações do documento** | |
| **Referência** | Breaking Crypto the Simple Way – Guilherme Gonsales de Sá |
| **N° Revisão** | 1 |
| **Data de publicação** | 04/05/2025 |
| **Link** | <https://tryhackme.com/room/breakingcryptothesimpleway> |

|  |  |  |
| --- | --- | --- |
| **Redação** | Guilherme Gonsales de Sá | Estudante |
| **Revisão** | Nome do revisor | Orientador |
| **Aprovação** | Nome do aprovador | Diretor |

|  |  |  |
| --- | --- | --- |
| **Histórico de revisões** | | |
| **N°** | **Entregas** | **Descrição** |
| **0** | 03/05/2025 | Produção |
| **1** | DD/MM/AAAA | Revisão |
| **2** | DD/MM/AAAA | Aprovação |

|  |  |
| --- | --- |
| **Informações do CTF** | |
| **Nível de Dificuldade** | Fácil |
| **Tipo de acesso** | Gratuito |
| **Conceitos envolvidos** | Criptografia |
| **Plataforma** | Tryhackme |
| **Área** | Red |

**Sumário**

[Contextualização 3](#_heading=h.gjdgxs)

[Desenvolvimento 3](#_heading=h.1fob9te)

[What is the flag? 3](#_heading=h.3znysh7)

[What is the secret used to encrypt the message? 4](#_heading=h.2et92p0)

[What is the flag? 5](#_heading=h.tyjcwt)

What is the flag? [8](#_heading=h.3dy6vkm)

[Conclusão 10](#_heading=h.1t3h5sf)

Scripts 10

[Referências 13](#_heading=h.4d34og8)

**Contextualização**

O CTF Breaking Crypto the Simple Way é um ctf inicial da área de criptografia, abrangindo alguns conceitos básicos sobre o tópico que devem ser compreendidos para aqueles que desejam se aprofundar na área. Embora ele possua conceitos de web, os mesmo não serão discutidos por serem basais e irrelevantes no que tange os objetivos do ctf.

**Desenvolvimento**

**What is the flag?**

A criptografia de uma chave RSA é dada pela premissa de impassibilidade computacional de chute. Ou seja, a premissa dessa criptografia é que, se a chave for grande o suficiente, não é possível encontra-la com chutes ou ataques de força bruta.

A encriptação RSA é um tipo de criptografia assimétrica, ou seja, existe um padrão de encriptação e um padrão de decriptação.

A encriptação via RSA é regida por

onde:

c = texto cifrado

m = mensagem original

e = expoente publico

n = módulo

Já a decriptação é regida por

onde d = expoente privado.

O módulo n é regido pela equação

onde p e q são dois números primos distintos.

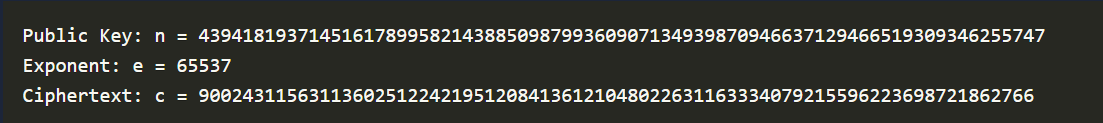
Entendamos os expoentes. O expoente público deve ser co-primo da função phi(n), regida por

onde phi é a função totiente de Euler.

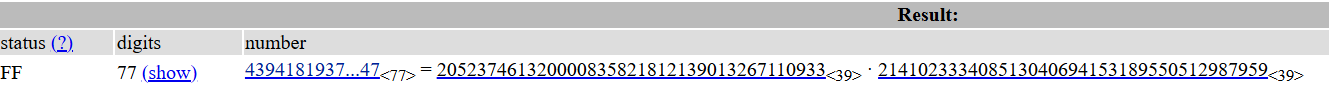
Via de regra, o expoente público é um número pequeno, como 65537 (ou (2^16)+1). Sabe-se que o expoente privado é relacionado ao expoente público, sendo ele regido por:

O termo ‘mod’ está sendo utilizado pois a aritmética feita é uma aritmética modular.

Com os esclarecimentos feitos, olhemos o exercício proposto. De início, temos as seguintes informações:



Já é possível identificar alguns dos elementos pivotais para decriptação da mensagem. O primeiro passo é fazer a fatoração de ‘n’ em ‘p’ e ‘q’. Para tal, pode-se confeccionar um script de força bruta para encontrar os fatores ou, visto que o número apresentado não é muito grande, utilizarmo-nos do site ‘FactorDb’, uma base de dados de fatoração em primos. Com a ajuda da database, temos:



Sendo p = 205237461320000835821812139013267110933

e b = 214102333408513040694153189550512987959

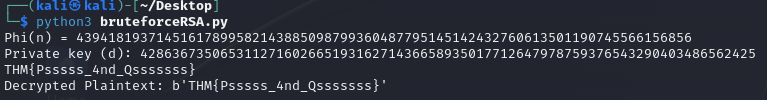
Com os fatores em mãos, pode-se criar um script em python que:

- Calcule a função totiente, phi(n)

- Calcule a chave privada

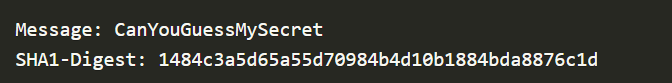
- Descriptografe a mensagem

Executando o script, tem-se:



Visto que a flag foi exposta, pode-se considerar o exercício concluído.

**What is the secret used to encrypt the message?**

De imediato, nos deparamos com o seguinte: 

Sabemos a mensagem e o tipo de criptogrfia utilizado. Antes de resolver, entendamos o que isso significa.

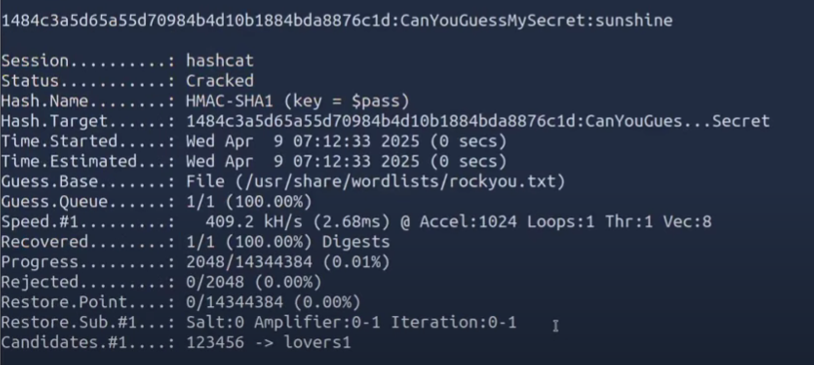
As criptografias de tipo SHA (Secure Hash Algorithym) são algoritmos que criptografam um texto, de qualquer tamanho de entrada, para uma saída de 160bits (em SHA1), ou 40 caracteres em hexadecimal (em SHA1). Cada entrada tem uma apenas uma saída, e não é possível reverter a saída apenas com ela mesma. Idealmente, todas as saídas são únicas (resistente a colisão).

Nosso objetivo é, portanto, descobrir a chave que foi usada para criptografar a mensagem.

Para isso, utilizemos a ferramenta hashcat, com o comando a seguir:



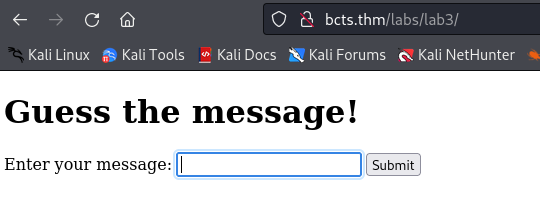
O conteúdo do arquivo ‘digest.txt’ é <hash>:<message>, que nos foi fornecido anteriormente. Dessa forma, o hashcat tentará encontrar uma chave, presente na wordlist rockyou.txt, que criptografe corretamente nossa mensagem para o hash encontrado. Dessa forma, temos:

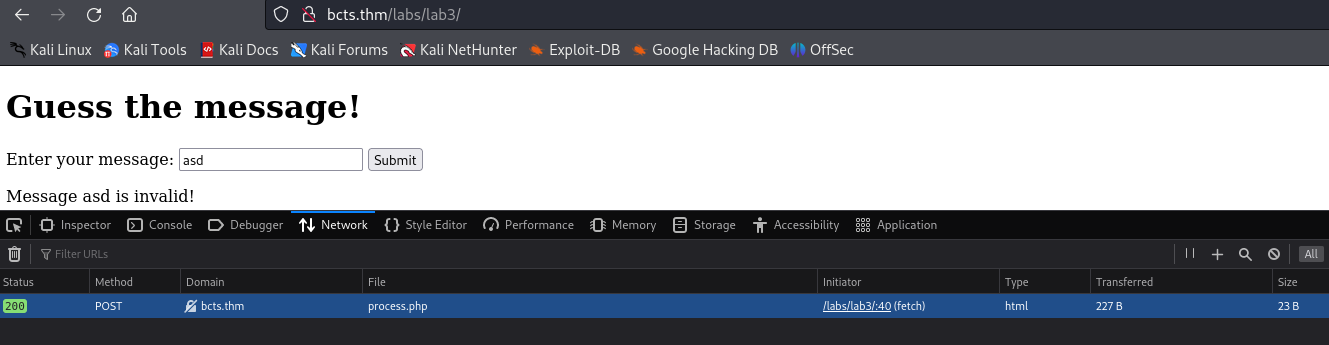


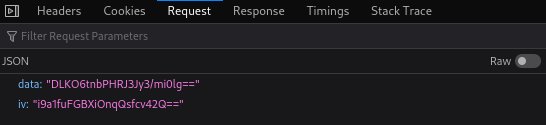
Observa-se que encontramos o resultado ‘sunshine’, sendo essa nossa chave e concluindo o exercício.

**What is the flag?**

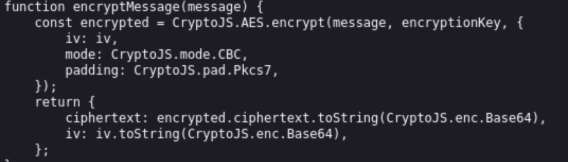
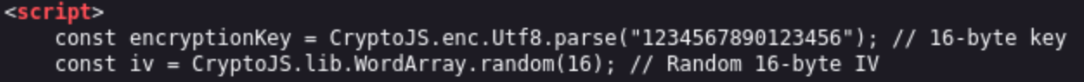
Tendo em vista o objetivo de descriptografar a mensagem utilizando uma chave criptográfica exposta, acessemos a URL fornecida pelo exercício para encontrá-la. Ao acessar a URL, tem-se:



Enviemos uma requisição qualquer para analisar como fica o front-end. Ao enviar uma requisição com conteúdo qualquer, analisemos a request:



Embora tenhamos alguns dados, não sabemos o tipo de criptografia utilizado. Para descobri-lo, analisemos o código mais a fundo.



Nos trechos acima, observa-se que encontramos uma clara confirmação da criptografia utilizada: AES

AES (Advanced Encryptio Standard) é um tipo de criptografia simétrica (mesma chave de encriptação e decriptação) que aceita chaves de 128, 192 e 256 bits. A mensagem é dividida em blocos de 16 bytes (128 bits) e comparada com a chave. No caso visto, encontramos um IV (Inicialization vector), que indica um protocolo CBC (Cipher Block Chaining), que defina a forma como os blocos estão conectados entre si.

Dessa forma, a criptografia usada é do tipo AES-CBC, onde, de maneira genérica, a encriptação é regida por

onde:

C1 = primeiro bloco de texto criptografado

P1 = primeiro bloco de texto puro

⊕ = operação XOR entre os bits

IV = Initialization vector, utilizado para fazer a primeira encriptação

Cn = n-ésimo bloco de texto criptografado

Pn = n-ésimo bloco de texto puro

Cn-1 = bloco de texto encriptado antecessor ao n-ésimo. Válido apenas a partir do segundo bloco de texto

Já a decriptação é regida de maneira genérica por

com a mesma limitação imposta em Cn-1.

Com essas equações em mãos, pode-se escrever um script que automatize esse processo. O script deve:

- Conectar-se ao servidor para averiguar a resposta

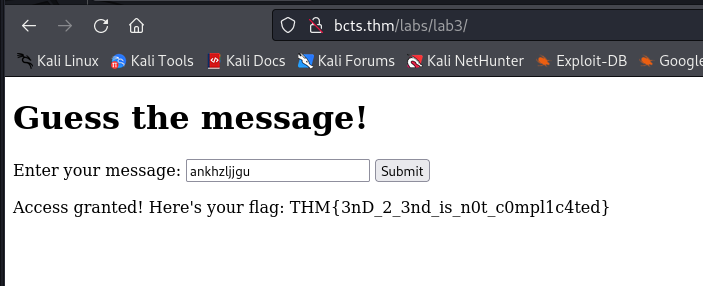
- Encriptar a mensagem (dada por uma wordlist)

- Parar no retorno de ‘Acess granted’, dado pelo servidor

Após a confecção de um script que cumpre tais requisitos, tem-se:



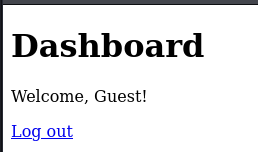
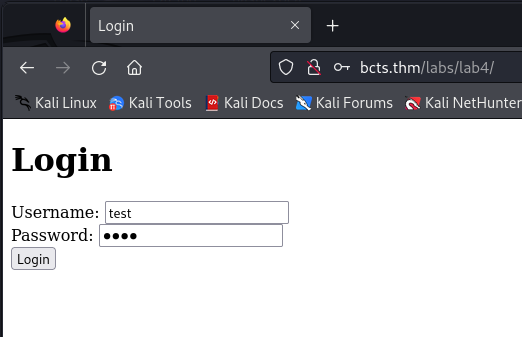
Isso significa que, ao enviar uma request com conteúdo ‘ankhzljjgu’ ao servidor, conseguiremos acesso ao mesmo e poderemos ver a flag. Para confirmação, façamos isso manualmente:



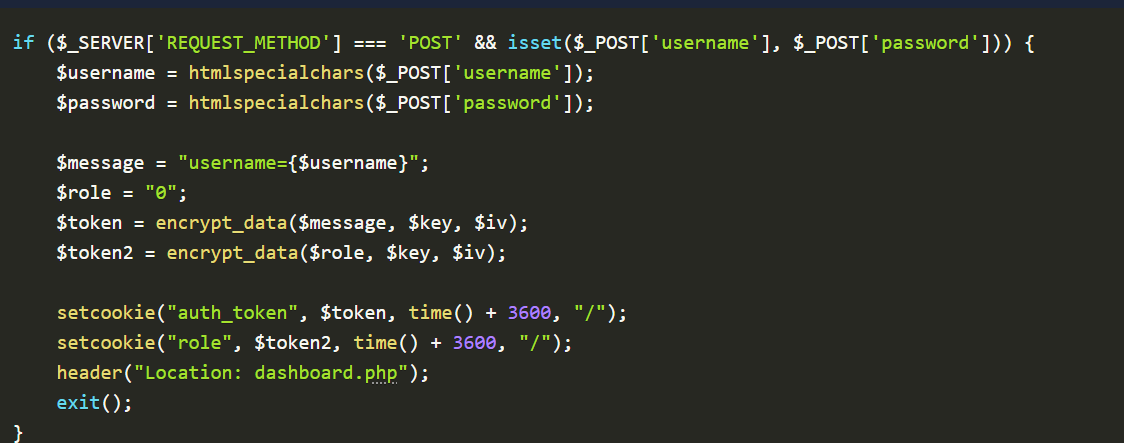
Como esperado, o script retornou corretamente a string que nos garantia acesso, concluindo o exercício.

**What is the flag?**

De início, acessemos a URL fornecida pelo exercício e, em sequência, coloquemos credenciais quaisquer para verificar o funcionamento da página



Observa-se que conseguimos um acesso, mesmo sem nos importar com as credencias. É seguro assumir que o acesso que obtivemos não possui privilégios elevados, o que provavelmente é necessário para obtermos a flag.

Ao inspecionar a página, obtemos: 

Observam-se dois atributos interessantes: a presença de um IV e a presença de ‘role’. Como discutido a priori, a aparição do IV indica que estamos lidando com uma criptografia do tipo AES-CBC, embora não tenhamos acesso ao IV, tampouco a chave. Isso elimina a possibilidade do ataque utilizado anteriormente.  
No entanto, a presença de ‘role’ se destaca. Ao analisar o código, perceb-se que role tem um valor definido que foi encriptado: 0. Isso indica a possibilidade de um ataque de bitflipping.

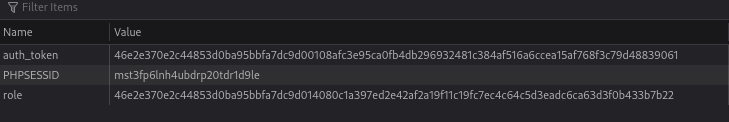
Além disso, percebemos que ele é armazenado em cookies, o que é um forte indício que ele pode ser manipulado.

Resta a pergunta: qual bit trocar?

No caso analisado, ao converter ‘0’ para ASCII, temos 0x30. Sabe-se também que ’1’, em ASCII, é 0x31.

Sendo assim, com o indício que apenas o numeral ‘0’ está sendo encriptado, deve-se altera-lo no IV, visto que podemos descobri-lo e alterar seu valor.

Ao analisar ‘cookies’, tem-se:



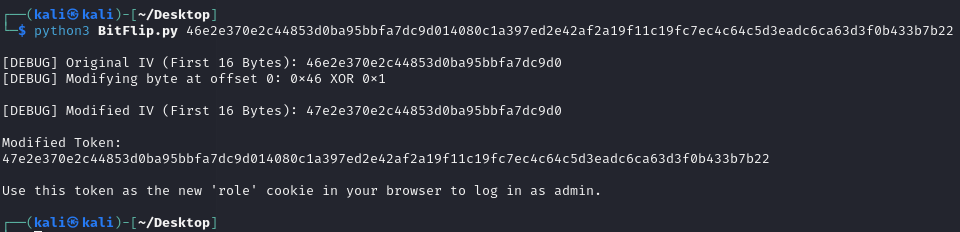
Agora que encontramos nosso cookie encriptado, devemos fazer um script que:

- Decodifique o cookie para encontrar o IV

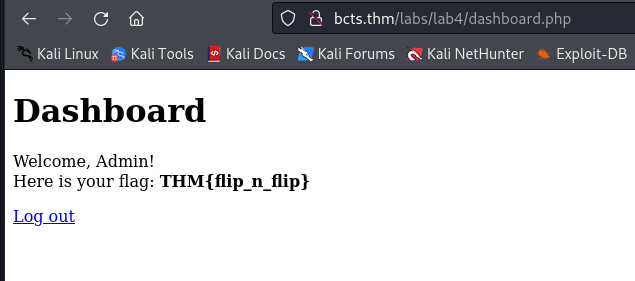
- Altere o valor do último bit do IV, para transformar ‘0’ em ‘1’

- Encripte novamente a mensagem para obtermos um cookie válido

Após criar o script e executa-lo, tem-se:



Agora que encontramos o ovo valor do cookie ‘role’, devemos altera-lo na página.

Ao alterar o cookie e recarregar a página, temos:

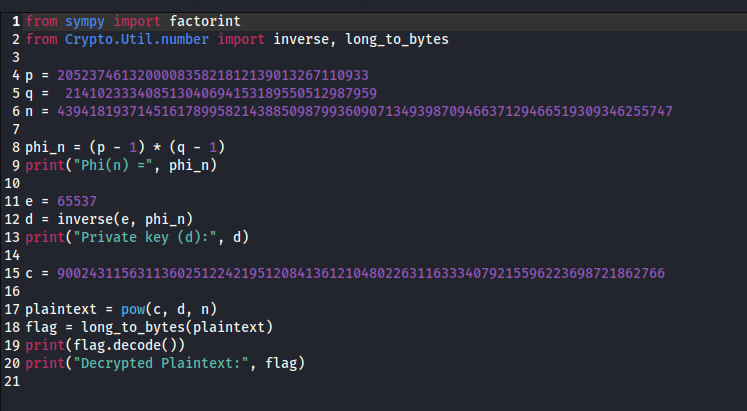
Dessa forma, conseguimos um login de administrador e encontramos a flag, concluindo o desafio.

**Conclusão**

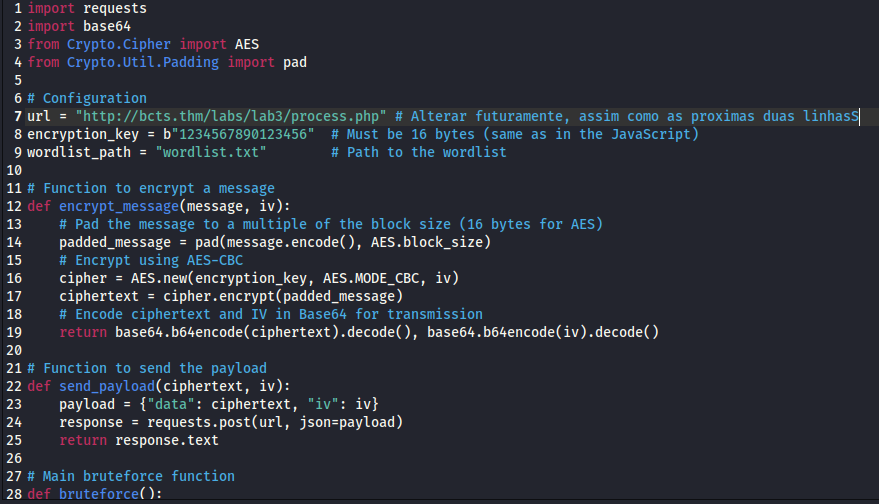
Apesar do ctf abordar apenas conceitos iniciais de criptografia, ele constrói uma sólida base para aqueles que querem se aprofundar na área de análise criptográfica, sendo uma excelente recapitulação dos fundamentos.

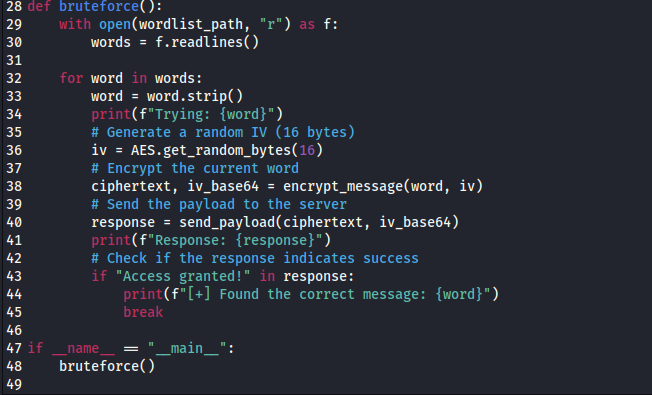
**Scripts**

Script 1 – Brute Force de chave RSA

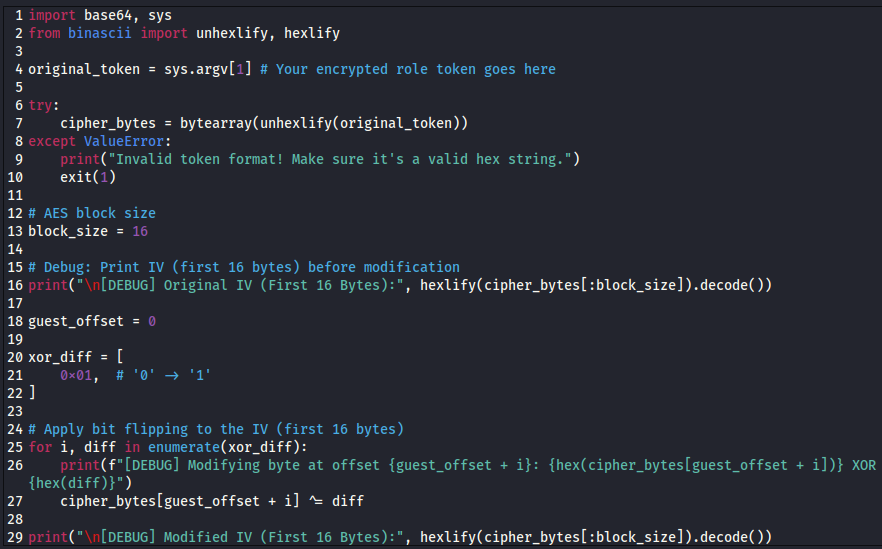


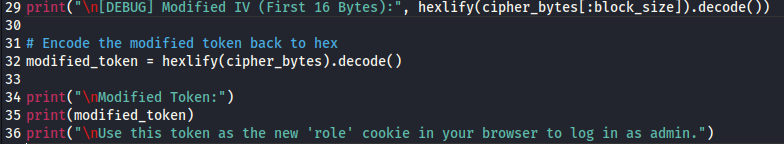
Script 2 – Decriptação de AES-CBC com chave exposta





Script 3 – Bit Flipper





**Referências**

<https://tryhackme.com/room/breakingcryptothesimpleway>

<http://www.factordb.com/>