EgoRain e EgoHash - Rainbow Table e Hash Cracking com AES-256-ECB

Citeforma | CET 8493 | UFCD 9195 Enquadramento operacional de cibersegurança | Formador: João Almeida

Hugo Miguel Félix Bugalho | João Rodrigo Mota da Costa | João Miguel Oliveira da Costa

Introdução

Neste documento, iremos documentar o EgoHash e o EgoRain, dois programas escritos em Python que criam uma *rainbow table* e um *hash cracker* com encriptação simétrica AES-256 em modo ECB. Mais uma vez, este trabalho foi feito de acordo com as instruções dadas pelo formador João Almeida.

Como funcionam?

EgoRain

Para este trabalho foi pedido que a *rainbow table* tivesse apenas palavras-passe com quatro caracteres. Após uma sessão de esclarecimento com o formador, entendemos que o processo de criação da mesma iria ser um pouco mais complexo do que o antecipado, visto que não existem livrarias nem esquemas para fazer algo deste género com este tipo de encriptação.

Chegamos então à conclusão que tinhamos de fazer a *rainbow table* sem automação, ou seja, criar um script em Python que gerasse uma lista de palavras-passe com quatro caracteres, estender as mesmas para ter 16 caracteres, escolher uma letra aleatória em cada bloco de quatro caracteres para criar uma nova palavra-passe e usar a palavra-passe de 16 caracteres como a chave de encriptação para a nova palavra-passe. Ao inicio, ficamos bastante confusos com as ideias, mas pusemos mãos à obra.

Para termos uma base com que trabalhar, pedimos a um modelo de AI para nos dar um exemplo de código, ao que o mesmo respondeu com código funcional, mas utilizando MD5 como o algoritmo para criar as nossas *hashes*. Com isto, tivemos uma ideia inicial de como o código iria fluir e qual seria a melhor ordem para cada processo até chegarmos ao nosso produto final.

Definimos os caracteres pedidos numa variável `charset` e utilizamos um método menos ortodoxo para a criação das palavras-passe, em que fizemos um conjunto de quatro loops `for` para

escolher cada letra do alfabeto até não haver mais hipóteses, gerando assim todas as palavras-passe possíveis com quatro caracteres. Estas são guardadas para uma variável `password` e vão para uma lista denominada `passwords`. Apesar de este método funcionar, gostariamos de ter feito de forma diferente, de modo a que houvesse mais controlo sobre a geração das palavras-passe, o que levaria a código mais optimizado e menos utilização de memória, que neste tipo de programas é considerado crucial.

Após a criação de uma palavra-passe, esta é repetida quatro vezes para gerar a palavra-passe de 16 caracteres, guardar na variável `passwords_16` e utilizamos a livraria `random` para escolher quatro caracteres aleatórios da palavra-passe de 16 caracteres para gerar uma nova palavra-passe de quatro caracteres. Esta é então guardada para uma nova variável `new_password` e vai para a lista `new_passwords`.

Seguindo a confusão da geração de palavras-passe, chegamos então à fase de encriptação. Este passo foi relativamente fácil utilizando o `import AES` da livraria `Crypto.Cipher`. Como mencionado anteriormente, utilizamos as palavras-passe de 16 caracteres como a chave de encriptação para cada `new_password` e guardamos a hash para uma lista `encrypted_passwords`.

Finalmente, para guardar as palavras-passe e as suas *hashes*, abrimos um ficheiro `rainbow_table.txt` e guardamos a palavra-passe em texto limpo, seguido da nova palavra-passe, seguido da *hash* em formato hexadecimal, tudo separado por dois pontos. Por exemplo, `{pass}:{nova_pass}:{hash}`.

EgoHash

Para o EgoHash, o código foi relativamente fácil de criar. Só foi necessário carregar a nossa lista de *hashes* e a nossa *rainbow_table* em memória, usar o `line.strip` e `line.split` para separar a *hash* e usar o loop `for` com um loop `if` para ver se cada *hash* no ficheiro `hashes.txt` batia com alguma *hash* dentro do ficheiro `rainbow_table.txt` e apresentar a palavra-passe em texto limpo.

Para dar um efeito mais cómico e para manter com o nosso tema "Ego", caracterizamos as mensagens a indicar se o programa encontrou ou não a palavra-passe com uma personagem do mesmo nome, que se dirige na terceira pessoa. Por exemplo, quando encontra uma *hash* correspondente, o programa diz `Ego encontrou {pass} para a hash {hash}.`. Caso não encontre nada, o mesmo diz `Ego não encontrou nada para a hash {hash}.`

Conclusão

Admitidamente, este trabalho foi interessante e extremamente desafiante. Houve alturas em que estavamos convencidos que seria impossível fazer este trabalho, mas com alguma motivação e pensamento critico, arranjamos uma maneira de ultrapassar o desafio e tornar o projeto algo mais único e personalizável.