Hash

Prof. Walter Priesnitz Filho, PhD. - walter@redes.ufsm.br

Visão Geral

- *One Way Hashes*
- *Cryptographic One-Way Hashes*
- *Message Authentication Codes*
- *Perfect Forward Secrecy*
- Exercícios

One-Way Hashes

- Uma função de *hash* é um algoritmo que produz uma saída de tamanho fixo a partir de uma entrada com dados de um comprimento arbitrário ou aleatório.
- O armazenamento de senhas como *hashes* oferece um certo nível de segurança caso o banco de dados de armazenamento de senhas seja comprometido.
- É importante observar que nem todos os algoritmos de *hash* são adequados para criptografia; aqueles que são, são chamados de funções de *hash* criptográficas.

Cryptographic One-Way Hashes

- Para que uma função de *hash* de sentido único seja usada em sistemas criptográficos, o algoritmo deve oferecer resistência à pré-imagem, resistência secundária e resistência à colisão:
  - Resistência à Colisão: é computacionalmente inviável encontrar quaisquer duas entradas distintas que mapeiem para a mesma saída.
  - Resistência à Pré-imagem: dada uma saída escolhida aleatoriamente, é computacionalmente inviável encontrar qualquer entrada que mapeie para essa saída.
  - Resistência Secundária: dado um valor de entrada, é computacionalmente inviável encontrar um segundo valor de entrada (distinto) que mapeie para a mesma saída que o primeiro valor.

Leitura complementar: https://csrc.nist.gov/glossary/term/cryptographic_hash_function

Birthday Attack

- A força da função de *hash* é, aproximadamente, a metade do comprimento do *hash* devido à probabilidade produzida pelo ataque de aniversário (*Birthday Attack*).
- Considere o cenário em que um professor com uma classe de 23 alunos (*n = 23*) pergunta a data de aniversário de todos para determinar se dois alunos têm a mesma data de aniversário.
  - Probabilidade que os aniversários sejam em dias diferentes

$$\bar{p}(n) = 1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{n-1}{365}\right) = \frac{365 \cdot 364 \cdots (365 - n + 1)}{365^n} = \frac{365!}{365^n (365 - n)!}$$

  - Probabilidade de os aniversários serem no mesmo dia

$$p(n) = 1 - \bar{p}(n).$$

Birthday Attack

- A força da função de *hash* é, aproximadamente, a metade do comprimento do *hash* devido à probabilidade produzida pelo ataque de aniversário (*Birthday Attack*).

n	p(n)
10	12%
20	41%
23	50,7%
30	70%
33	**77,497185417572%**
50	97%
100	99,99996%

Birthday Attack

```
def birthday(n):
    p = (1.0/365)**n
    for i in range((366-n),366):
        p *= i
    return 1-p

print(birthday(33))
```

Message Authentication Codes

- É um algoritmo de resumo/síntese de mensagens baseado em chave que pode ser usado para verificar a integridade da mensagem, para verificar a autenticidade do remetente da mensagem ou ambos.
- O HMAC foi amplamente adotado para uso em vários sistemas e domínios, como comunicações entre servidores, APIs de serviços da Web etc.
  - Um uso bem conhecido do HMAC é nas chamadas de API do AWS da Amazon, em que a assinatura é gerada usando o HMAC.
- O HMAC pode usar uma variedade de algoritmos de *hash*, como MD5, SHA1, SHA256, etc.
- A função HMAC não exige muito processamento, por isso tem sido amplamente aceita e é fácil de implementar em dispositivos móveis e incorporados, mantendo um bom nível de segurança. O exemplo de código a seguir mostra como gerar um resumo HMAC-MD5 com Python:

Message Authentication Codes

```
import hmac
from hashlib import md5
key = b'DECLARATION'
h = hmac.new(key,b'',md5)
# adiciona o conteúdo da mensagem
h.update(b'Teste de HMAC!!!!')
# imprime a assinatura/hash HMAC
print (h.hexdigest())
```

Perfect Forward Secrecy

- Também conhecido como *Forward Secrecy* (FS), é um conjunto de protocolos de acordo de chaves que dá aos participantes da troca de mensagens a garantia de que suas mensagens não serão comprometidas, mesmo que a chave privada do servidor seja comprometida.
- Protege sessões criptográficas anteriores contra futuros comprometimentos de senhas ou chaves secretas.
- O comprometimento de uma única chave de sessão não afetará nenhum dado além dos trocados na sessão específica, pois gera uma chave de sessão exclusiva para cada sessão individual;
  - O OpenSSL foi afetado pela exploração Heartbleed*.
    - As comunicações criptografadas e as sessões registradas que possam ter sido comprometidas não poderão ser usadas para descriptografar comunicações futuras.

* https://bit.ly/heartBleed

Perfect Forward Secrecy

- O PFS está presente em várias implementações de protocolo, como IPSec (opcional), SSH, STARTTLS, ATS, …
- Desde o final de 2011, o Google fornece FS com TLS por padrão para os usuários de seus serviços com Gmail, Google Docs e serviços de pesquisa criptografada.
- O Facebook, desde maio de 2014, oferece suporte ao STARTTLS. O TLS 1.3 foi lançado em agosto de 2018.
- O *App Transport Security* (ATS) usado pela Apple em aplicativos iOS, e se tornou obrigatório em todos os seus dispositivos iOS após 1º de janeiro de 2017.

Exercícios

1)Dado o código abaixo, que cifra uma mensagem em Python, escreva o código para decifrar esta mensagem.

```
from cryptography.fernet import Fernet
chave = Fernet.generate_key()
cipher_suite = Fernet(chave)
mensagem_cifrada = cipher_suite.encrypt(b'Minha mensagem secreta')
```

2)Escreva um programa em Python que:

a)Calcula o hash MD5 de uma string

b)Calcula o hash SHA-256 de uma string

3)O que é um ataque de força bruta?

4)Escreva um programa em Python que faça um ataque de força bruta sobre o hash (MD5 - ad9be0b5d43f9e2aba895f3ede723aa1) deste PIN de quatro dígitos.