OpenSSL

Aluno: Gustavo Emanuel Kundlatsch

a) Quais os os comandos que o OpenSSL oferece? De uma visão geral dos comandos, sem entrar em detalhes sobre parâmetros;

asn1parse: Analisa uma sequência ASN.1.
ca: Gerenciamento da Autoridade de Certificação (CA).
ciphers: Determinação da descrição de cipher suite.
cms: Utilitário CMS (Cryptographic Message Syntax).
crl: Gerenciamento da lista de revogação de certificados (CRL).
crl2pkcs7: Conversão de CRL para PKCS # 7.
dgst: Cálculo do digest da mensagem.
dh: Gerenciamento de parâmetros Diffie-Hellman. Obsoleto por dhparam (1).
dhparam: Geração e gerenciamento de parâmetros Diffie-Hellman. Substituído por genpkey (1) e pkeyparam (1).
dsa: Gerenciamento de dados DSA.
dsaparam: Geração e gerenciamento de parâmetros de DSA. Substituído por genpkey (1) e pkeyparam (1).
ec: Processamento de chaves EC (curva elíptica).
ecparam: Manipulação e geração de parâmetros EC.
enc: codificação com cifras.
engine: Informações e manipulação de engine (módulo carregável).
errstr: Conversão de número de erro para string.
gendh: Geração de parâmetros Diffie-Hellman. Obsoleto por dhparam (1).
gendsa: Geração de chave privada DSA a partir de parâmetros. Substituído por genpkey (1) e pkey (1).
genpkey: Geração de chave privada ou parâmetros.
genrsa: Geração de chave privada RSA. Substituído por genpkey (1).
nseq: Crie ou examine uma sequência de certificados Netscape.
ocsp: Utilitário de protocolo de status de certificado online.
passwd: Geração de senhas com hash.
pkcs12: Gerenciamento de dados PKCS # 12.
pkcs7: Gerenciamento de dados PKCS # 7.
pkcs8: Ferramenta de conversão de chave privada no formato PKCS # 8.
pkey: Gerenciamento de chaves públicas e privadas.
pkeyparam: Gerenciamento de parâmetro de algoritmo de chave pública.
pkeyutl: Utilitário de operação criptográfica de algoritmo de chave pública.
prime: Calculo de números primos.
rand: Gerar bytes pseudo-aleatórios.
rehash: Crie links simbólicos para arquivos de certificado e CRL nomeados pelos valores de hash.
rsa: Gerenciamento de chaves RSA.
rsautl: Utilitário RSA para assinatura, verificação, criptografia e descriptografia. Substituído por pkeyutl (1).

s_client: Implementa um cliente SSL / TLS genérico que pode estabelecer uma conexão transparente com um servidor remoto SSL / TLS. Destina-se apenas a fins de teste e fornece apenas funcionalidade de interface rudimentar, mas internamente usa quase todas as funcionalidades da biblioteca OpenSSL.
s_server: Implementa um servidor SSL / TLS genérico que aceita conexões de clientes remotos que SSL / TLS. Destina-se apenas a fins de teste e fornece apenas uma interface rudimentar funcionalidade, mas internamente usa quase todas as funcionalidades da biblioteca OpenSSL. Ele fornece um protocolo próprio orientado a linha de comando para testar funções SSL e um protocolo HTTP simples recurso de resposta para emular um servidor da web compatível com SSL / TLS.
s_time: temporizador de conexão SSL.
sess_id: Gerenciamento de dados de sessão SSL.
smime: Processamento de email S/MIME.
speed: Medição da velocidade do algoritmo.
spkac: Utilitário de impressão e geração de SPKAC.
srp: Gerenciar o arquivo de senha SRP.
storeutl: Utilitário para listar e exibir certificados, chaves, CRLs, etc.
ts: ferramenta de Time Stamping Authority (cliente / servidor).
verify: Verificação do certificado X.509.
version: Informações sobre a versão do OpenSSL.
x509: Gerenciamento de dados do certificado X.509.

Comandos de message digest: blake2b512, blake2s256, md2, md4, md5, mdc2, rmd160, sha1, sha224, sha256, sha384, sha512, sha3-224, sha3-256, sha3-384, sha3-512, shake128, shake256 e sm3.

Comandos de codificação e sifras: aes128, aes-128-cbc, aes-128-cfb, aes-128-ctr, aes-128-ecb, aes-128-ofb, aes192, aes-192-cbc, aes-192-cfb, aes-192-ctr, aes-192-ecb, aes-192-ofb, aes256, aes-256-cbc, aes-256-cfb, aes-256-ctr, aes-256-ecb, aes-256-ofb, aria128, aria-128-cbc, aria-128-cfb, aria-128-ctr, aria-128-ecb, aria-128-ofb, aria192, aria-192-cbc, aria-192-cfb, aria-192-ctr, aria-192-ecb, aria-192-ofb, aria256, aria-256-cbc, aria-256-cfb, aria-256-ctr, aria-256-ecb, aria-256-ofb, base64, bf, bf-cbc, bf-cfb, bf-ecb, bf-ofb, camellia128, camellia-128-cbc, camellia-128-cfb, camellia-128-ctr, camellia-128-ecb, camellia-128-ofb, camellia192, camellia-192-cbc, camellia-192-cfb, camellia-192-ctr, camellia-192-ecb, camellia-192-ofb, camellia256, camellia-256-cbc, camellia-256-cfb, camellia-256-ctr, camellia-256-ecb, camellia-256-ofb, cast, cast-cbc, cast5-cbc, cast5-cfb, cast5-ecb, cast5-ofb, chacha20, des, des-cbc, des-cfb, des-ecb, des-ede, des-ede-cbc, des-ede-cfb, des-ede-ofb, des-ofb, des3, desx, des-ede3, des-ede3-cbc, des-ede3-cfb, des-ede3-ofb, idea, idea-cbc, idea-cfb, idea-ecb, idea-ofb, rc2, rc2-cbc, rc2-cfb, rc2-ecb, rc2-ofb, rc4, rc5, rc5-cbc, rc5-cfb, rc5-ecb, rc5-ofb, seed, seed-cbc, seed-cfb, seed-ecb, seed-ofb, sm4, sm4-cbc, sm4-cfb, sm4-ctr, sm4-ecb e sm4-ofb.

Comandos extraídos do manual do OpenSSL ($ man openssl).

b) Quais algoritmos de criptografia simétrica são disponibilizados (veja em detalhes o comando "openssl-enc")? E desses, quais modos de operação?

Algoritmo	**Modos**
Base 64	-
Blowfish	CBC, CFB, ECB e OFB
CAST	CBC
CAST5	CBC, CFB, ECB e OFB
ChaCha20	-
DES	CBC, CFB, ECB e OFB
Two key triple DES EDE	CBC, CFB, ECB e OFB
Three key triple DES EDE	CBC, CFB, ECB e OFB
desx	-
GOST 28147-89	CFB e CNT
IDEA	CBC, CFB, ECB e OFB
128 bit RC2	CBC, CFB, ECB e OFB
64 bit RC2	CBC
40 bit RC2	CBC
128 bit RC4	-
64 bit RC4	-
40 bit RC4	-
RC5	CBC, CFB, ECB e OFB
SEED	CBC, CFB, ECB e OFB
SM4	CBC, CFB, CTR, ECB e OFB
128/192/256 bit AES	CBC, CFB 128/8/1 bit, CTR, ECB e OFB
128/192/256 bit ARIA	CBC, CFB 128/8/1 bit, CTR, ECB e OFB
128/192/256 bit Camellia	CBC, CFB 128/8/1 bit, CTR, ECB e OFB

c) Como são definidas (estabelecidas) as senhas e as chaves desses algoritmos?

Vários comandos OpenSSL aceitam argumentos de senha, normalmente usando -passin e -passout para senhas de entrada e saída, respectivamente. Isso permite que a senha seja obtida de várias fontes. Ambas as opções aceitam um único argumento, que pode ser a senha passada diretamente como argumento, uma variável de ambiente, o caminho para um arquivo que contém a senha, a leitura direta do stdin ou o número de um file descriptor. Se nenhum argumento de senha for fornecido e uma senha for exigida, o usuário é solicitado a inserir uma: normalmente, ela será lida no terminal atual sem mostrar o que está sendo digitado.
As chaves, por sua vez, são derivadas da senha que foi passada. Um método específico pode ser definido (de acordo com os digest disponíveis) ou o default será usado. Além disso, é possível definir o número de iterações para gerar a chave e se o salt será ou não utilizado.

d) O que é e para que é usado o "salt"?

O salt é uma cadeia utilizada para gerar a chave juntamente com a senha. Para evitar que a mesma chave seja gerada sempre que uma mesma senha é utilizada, a senha é combinada com uma cadeia chamada “salt”. O crypt(3) original, de 1970, define o salt como: “uma string de dois caracteres escolhida no conjunto [a-zA-Z0-9./]. Esta string é usada para perturbar o algoritmo de uma das 4096 maneiras diferentes”. O uso do salt não é obrigatório, mas é altamente recomendado (a não ser para propósitos de teste ou educacionais).

e) Combine uma senha com um colega seu, cifre e envie para ele um arquivo cifrado para ele. Veja se ele conseguiu decifrar? Como ele sabe qual é o algoritmo simétrico e o modo de operação que ele deveria usar para decifrar este arquivo? Onde fica esta informação?

Pelo arquivo criptografado não conseguimos descobrir o algoritmo usado. Para descriptografar, usamos um script para testar as diferentes opções de cifradores do OpenSSL. Foi preciso também passar a senha utilizada para gerar a chave.

f) Escolha um arquivo relativamente grande (alguns megabytes). Cifre-o. Analise o tamanho do arquivo cifrado em relação ao não cifrado. Explique a diferença no tamanho dos arquivos.

Comando utilizado:

```
taguma@lis:~$ openssl des  -e -in Downloads/12600.backup -v -out test.txt
```

Saída:

```
taguma@lis:~$ openssl des  -e -in Downloads/12600.backup -v -out test.txt
bufsize=8192
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bytes read   : 949309344
bytes written: 949309368
```

Leitura das primeiras linhas do arquivo criptografado:

```
taguma@lis:~$ sed -n 1,3p test.txt
Salted__[illegible]H[illegible]a
[illegible]n[illegible]?[illegible]
o][illegible]7[illegible]%S[illegible]lfNSLgT'[illegible]n[illegible]Y[illegible]1[illegible]|2
[illegible]M+q[illegible]W[illegible]?.o[illegible]{[illegible]A:[illegible]4G7[illegible]h[illegible]t[illegible]8q[illegible]v[illegible]_[illegible]Q1[illegible],[illegible]a[illegible]
}a/[illegible]/Z[illegible]AZ[illegible]Y[illegible]Z[illegible]q
[illegible]
gaf[illegible]ci[illegible]Vc[illegible]4[illegible]!O;[illegible]=[illegible]z2n[illegible]
```

A adição do cabeçalho (que informa que foi usado o salt) no arquivo parece ter causado a diferença de tamanho.

g) Qual é o significado de "engine" para o OpenSSL?

Uma engine é um objeto implementado para acelerar o processamento de operações criptográficas. Uma engine normalmente é implementada em hardware, e se integra com as funções do OpenSSL através de suas funções built-in.