



Sicurezza II A.A. 2010-2011

OpenSSL: Introduzione all'uso



Speaker:

André Panisson, PhD student

Università degli Studi di Torino, Computer Science Department

Corso Svizzera, 185 - 10149, Torino, Italy

panisson@di.unito.it



Introduzione

- o **OpenSSL** è un progetto **open source** che ha lo scopo di sviluppare una libreria portabile e multi-piattaforma che offra le seguenti funzionalità:
 - SSL (Secure Sockets Layer) versione 2 e 3
 - TLS (Transport Layer Security) versione 1
 - Algoritmi di crittografia simmetrica e asimmetrica
- o Come per la maggior parte dei progetti open source, esiste una comunità su base internazionale che si occupa di svilupparla
- o La licenza di OpenSSL ne permette l'uso e la redistribuzione, anche a scopi commerciali



Descrizione

- o OpenSSL è composta di tre parti distinte:
 - **libssl.a**: libreria che contiene l'implementazione e le funzioni utili al corretto funzionamento di SSLv2, SSLv3 e TLSv1, lato client o server
 - **libcrypto.a**: libreria che contiene le funzioni crittografiche e le funzioni di gestione dei certificati; più dettagliatamente:
 - cifrari simmetrici: DES, 3DES, RC2, RC4, Blowfish, AES, ... nelle modalità ecb, cbc, cfb e ofb
 - cifrari asimmetrici: RSA, DSA, Diffie-Hellman
 - funzioni di HASH: MD2, MD5, SHA-1, MDC2, ...
 - funzioni per la lettura, creazione e gestione di certificati in formato X509 (PEM, DER, ...); funzioni per la gestione di una Certification Authority.
 - ...
 - **openssl**: un tool a linea di comando che permette di utilizzare la maggior parte delle funzioni presenti in libcrypto.a



Download e installazione



- o Download dei sorgenti: <http://www.openssl.org/>
- o A oggi, l'ultima versione disponibile è la **1.0.0d**, ma la versione installata sul laboratorio è la **1.0.0b**
- o Dopo aver scaricato i sorgenti è necessario scompattarli:
 - **gunzip openssl-1.0.0d.tar.gz**
 - **tar xvf openssl-1.0.0d.tar**
 - **(mv openssl-1.0.0d /tmp)**
 - **(cd /tmp/openssl-1.0.0d)**
- o Successivamente è possibile iniziare la procedura di compilazione descritta nel file INSTALL



Installazione

```
$ ./config
```

```
$ make
```

```
$ make test
```

- Al termine dell'esecuzione del make, dovrete poter eseguire correttamente il comando

```
openssl-1.0.0d/apps/openssl
```

- L'esecuzione del comando provoca l'apertura di una shell
 - Digitare **?** per ottenere l'elenco delle istruzioni
 - Provare ad esempio il comando **rand**
 - **rand -base64 100**



OpenSSL da riga di comando

- Per accedere alla shell, basta eseguire il comando **openssl**
- Si noti che tutti i comandi che è possibile inviare tramite la shell di openssl, possono essere utilizzati anche come parametro di openssl stesso:
 - **OpenSSL> rand -base64 100**
 - **./openssl rand -base64 100**
- È possibile trovare la documentazione sull'utilizzo di OpenSSL da linea di comando sul sito ufficiale di OpenSSL (si noti che la documentazione non è sempre aggiornata):
<http://www.openssl.org/docs/apps/openssl.html>
- **man openssl**



Script da shell

- o Per eseguire gli esercizi proposti è necessario saper scrivere semplici **script di shell**
- o Nel seguente esempio, prepariamo uno script che calcola 2 stringhe random rispettivamente lunghe 100 e 200 byte

- o **vi test.sh**
 - **#!/bin/bash**
 -
 - **openssl rand -base64 100**
 - **openssl rand -base64 200**
- o **chmod +x test.sh**
- o **./test.sh**



Esercizi preparatori (1)

- Preparare uno script che calcoli l'hash MD5, SHA1 e Ripemd160 di un file (comando **dgst**)
- Preparare uno script che esegua la cifratura simmetrica di un file con AES256, blowfish e Triple-DES in modalità CBC (comando **enc**)
- Preparare uno script che decifri i file cifrati al passo precedente (comando **enc**)
- Preparare uno script che calcoli il base64 di un file in input e che decodifichi un file codificato con base64 (comando **enc**)



Esercizi preparatori (2)

- Preparare uno script che:
 - generi una chiave privata RSA da 2048 bit (comando **genrsa**).
 - generi la corrispondente chiave pubblica (comando **rsa -pubout**).
 - stampi a schermo le informazioni sulle chiavi (comando **rsa -text**)
- Preparare uno script che esegua la cifratura e la decifratura con RSA utilizzando le chiavi precedentemente generate (comando **rsaut1**)
- Preparare uno script che esegua la firma e la verifica della stessa con RSA utilizzando le chiavi precedentemente generate (comando **rsaut1**)



PKI con OpenSSL

- o OpenSSL mette a disposizione una serie di metodi per creare e gestire una **Public Key Infrastructure**
- o I certificati emessi con le PKI create con OpenSSL rispettano lo standard **X.509**
- o Il primo passo per creare la propria PKI consiste nel creare il certificato relativo alla root CA (Certificate Authority)
- o Più precisamente è necessario creare una coppia di chiavi ed un certificato relativo alla chiave pubblica: **certificato autofirmato**
- o Successivamente è possibile emettere certificati per server e persone



Creazione del certificato della CA

- È necessario creare la chiave privata relativa alla **CA** (comando **genrsa**)
 - Salvare la chiave nel file **CAkey.pem**.
- Successivamente è necessario creare il certificato di chiave pubblica relativo alla CA; il comando da utilizzare è **req**
- I parametri da utilizzare sono i seguenti:
 - **-config ~/openssl-1.0.0d/apps/openssl.cnf**: indica ad OpenSSL dov'è il file di configurazione di default
 - **-key CAkey.pem**: indica quale chiave privata utilizzare per la generazione del certificato
 - **-new**: indica che si tratta di un nuovo certificato
 - **-x509**: indica di generare un certificato in formato **x509**
 - **-days 365**: indica che il certificato avrà validità per 365 giorni
 - **-out CAcert.pem**: indica in quale file salvare il certificato



Creazione del certificato della CA

- o Dopo aver lanciato il comando appena indicato verranno richiesti i dati relativi al certificato che si intende creare:

Country Name (2 letter code): *IT*
State or Province Name (full name): *Torino*
Locality Name (eg, city): *Torino*
Organization Name (eg, company): *Università di Torino*
Organizational Unit Name (eg, section): *DipInfo*
Common Name (eg, YOUR name): *Andre Panisson*
Email Address: *panisson@di.unito.it*



Creazione del certificato della CA

- Visualizzare il certificato appena creato (comando `x509`)
- Installare il certificato nel browser e visualizzarlo con il browser stesso
- **È fondamentale proteggere la chiave privata della CA, cifrandola con un algoritmo simmetrico (comando `rsa`)**



Procedura di emissione dei certificati

- La procedura di emissione dei certificati è divisa in più fasi:
 - generazione della chiave privata
 - generazione della richiesta di certificato
 - generazione del certificato
- Dei tre passi appena indicati, solo l'ultimo in genere è a carico della CA
- L'utente che desidera ottenere un certificato crea la propria chiave privata e successivamente crea una richiesta di certificato che contiene:
 - La chiave pubblica da certificare
 - I dati dell'utente che richiede il certificato
- La richiesta di certificato viene firmata con la chiave privata del richiedente
 - Questo permette alla CA di essere certa che l'utente sia in possesso della chiave privata corrispondente alla chiave pubblica che sta certificando



Procedura di emissione dei certificati

- La generazione della chiave avviene come abbiamo già visto in precedenza (comando **genrsa**)
- La generazione della richiesta avviene tramite il comando **req** e prevede i seguenti parametri:
 - **-config ~/openssl-1.0.0d/apps/openssl.cnf**
 - **-key newkey.pem**
 - **-new**
 - **-out newreq.pem**
- Si noti che, a differenza dei parametri utilizzati in precedenza, non sono presenti le opzioni **-x509 -days ###**
 - questo dipende dal fatto che si sta generando una richiesta di certificato e non un certificato



Procedura di emissione dei certificati

- Durante la generazione della richiesta, viene richiesto all'utente di specificare i dati relativi al certificato. Due dei parametri richiesti hanno un significato particolare:
 - **Common Name (eg, YOUR name) []**
 - il Common Name (CN) è particolarmente significativo quando si intende generare un certificato per un server che utilizza SSL (https, imaps, pops, ...). Per evitare che ci siano errori durante la comunicazione SSL è necessario che il CN corrisponda al nome DNS del server che si sta contattando.
 - Provare a contattare <https://www.educ.di.unito.it> e <https://goofy.educ.di.unito.it>.
 - Provare a visualizzare i dati relativi al certificato SSL utilizzato.
 - **Email Address []**
 - questo parametro è particolarmente significativo per i certificati che vengono rilasciati alle persone fisiche; infatti deve corrispondere al reale indirizzo di posta elettronica della persona, per evitare che ci siano errori legati a incongruenze fra mittente delle e-mail firmate e indirizzo indicato nel certificato.



Procedura di emissione dei certificati

- Durante la generazione della richiesta viene richiesto all'utente se vuole utilizzare una password per cifrare la richiesta stessa
 - in questo caso l'amministratore della CA, per generare il certificato, dovrà ricevere sia la richiesta di certificato che la password in questione
- Una volta generata la richiesta è possibile rilasciare il certificato utilizzando la CA; l'operazione avviene tramite l'utilizzo del comando x509 e dei seguenti parametri:
 - **-days ###**: indica per quanti giorni il certificato sarà valido
 - **-CA CAcert.pem**: indica il file del certificato della CA
 - **-CAkey CAkey.pem**: indica il file della chiave privata della CA
 - **-CAcreateserial**: indica che è necessario creare il numero seriale per il certificato - **UTILIZZARE SOLO AL PRIMO RILASCIO**
 - **-CAserial ca.srl**: indica il file che contiene il seriale da ultimo certificato rilasciato
 - **-req**: indica che il file in input è una richiesta di certificato
 - **-in newreq.pem**: indica il file della richiesta
 - **-out newcert.pem**: indica il file in cui salvare il certificato



Verifica dei certificati emessi

- o Utilizzare il comando **x509** per visualizzare il certificato appena emesso
- o Provare ad aprire il certificato appena emesso con il browser e controllare il rapporto fra il certificato della CA e quello appena emesso



Utilizzo di un certificato con la mail

- I client di posta si aspettano in input dei file in formato PKCS12
 - questo formato è utile per salvare in un unico file il certificato di chiave pubblica e la chiave privata cifrata
- OpenSSL permette la generazione dei file in questo formato con il comando **pkcs12**
pkcs12 -in cert.pem -inkey private.key
-export -out cert.pk12
- Il file così generato può essere importato nei repository del client di posta e/o del browser
- A questo punto è possibile inviare mail firmate
- Per inviare mail cifrate è necessario possedere il certificato del destinatario della mail



Creazione di un programma che usi la libreria

- Come abbiamo già detto, OpenSSL fornisce anche due librerie in C:
 - **libssl.a**: funzioni relative a SSLv2, SSLv3 e TLSv1;
 - **libcrypto.a**: funzioni crittografiche e le funzioni di gestione dei certificati.
- È possibile quindi scrivere programmi in C/C++ che utilizzino questa libreria per eseguire operazioni crittografiche di vario tipo.
- La stesura dei programmi per questa parte di esercitazione può essere fatta indifferentemente utilizzando i linguaggi C e C++.



Makefile

- Il modo più semplice per compilare un programma che utilizzi le librerie OpenSSL è quello di preparare un opportuno **makefile** per la compilazione.
- I parametri che devono essere presenti nel makefile sono:
 - **CC=gcc/g++**: nome del compilatore da usare
 - **LIBS=-L\$(HOME)/openssl-1.0.0d/ -lcrypto -lssl -ldl**: librerie che devono essere linkate e path verso le medesime
 - **INCLUDE=-I\$(HOME)/openssl-1.0.0d/include/**: path degli header file
 - **CFLAGS=-Wall -O2 \$(INCLUDE)**: flag del compilatore



Makefile

- o Un esempio di makefile funzionante per il file test.c è il seguente:

```
CC=gcc
TARGET=test
OBJS=test.o
LIBS=-L$(HOME)/openssl-1.0.0d/ -lcrypto -lssl -ldl
INCLUDE=-I$(HOME)/openssl-1.0.0d/include/
CFLAGS=-Wall -O2 $(INCLUDE)

all: $(OBJS)
    gcc $(CFLAGS) -o $(TARGET) $(OBJS) $(LIBS)

clean:
    rm -f $(OBJS) $(TARGET) *~ *.log
```



Primo programma di test

- o Come primo programma di test per OpenSSL, per controllare che la compilazione vada a buon fine, si può usare un semplice programma che inizializza l'ambiente di lavoro:

```
#include <stdio.h>
#include <stdlib.h>
#include "openssl/rsa.h"
#include "openssl/x509.h"
#include "openssl/evp.h"
#include "openssl/pem.h"
#include "openssl/err.h"

int main ()
{
    SSLeay_add_all_algorithms();
    ERR_load_crypto_strings();

    return 0;
}
```



Esercitazioni

- o Scrivere un programma che esegua la cifratura e la decifratura simmetrica di un file in modalità CBC.
- o La documentazione di [OpenSSL](#) e [Google](#) sono vostri amici!
- o Nel lucido seguente viene riportato lo scheletro del programma



Cifratura/decifratura in modalità CBC

```
int main (...)  
{  
    ...  
    FILE *src=fopen("chiaro", "rb");  
    FILE *dst=fopen("cifrato", "wb");  
    ...  
    EVP_CIPHER_CTX ctx;  
    EVP_CIPHER_CTX_init(&ctx);  
    EVP_CipherInit_ex(&ctx, EVP_XXX_cbc(), NULL, NULL, NULL, 1);  
    EVP_CIPHER_CTX_set_key_length(&ctx, keylen);  
    EVP_CipherInit_ex(&ctx, NULL, NULL, key, vect, 1);  
  
    while (num_read=fread(srcbuf, 1, 1024, src)) {  
        EVP_CipherUpdate(&ctx, dstbuf, &num_write, srcbuf, num_read);  
        fwrite(dstbuf, 1, num_write, dst);  
    }  
  
    EVP_CipherFinal_ex(&ctx, outbuf, &outlen);  
    EVP_CIPHER_CTX_cleanup(&ctx);}
```



Sicurezza II A.A. 2009-2010

OpenSSL: Introduzione all'uso

Grazie per l'attenzione!



Speaker:

André Panisson, PhD student

Università degli Studi di Torino, Computer Science Department

Corso Svizzera, 185 - 10149, Torino, Italy

panisson@di.unito.it



Queste slides sono state liberamente tratte dalle slides su *Computer Networks and Security* di *Michele Miraglia*, al quale va un ringraziamento speciale per il suo lavoro

©2011 by André Panisson. Permission to make digital or hard copies of part or all of this material is currently granted without fee *provided that copies are made only for personal or classroom use, are not distributed for profit or commercial advantage, and that new copies bear this notice and the full citation.*