

SSL, TLS ed OpenSSL

Ivan Visconti
Département d'Informatique
École Normale Supérieure
ivan.visconti@ens.fr

Overview

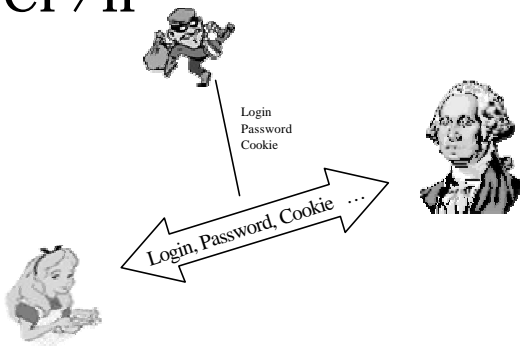
PARTE I: Il protocollo SSL

- » Motivazioni
- » SSL: l'handshake ed il record layer
- » Il protocollo TLS
- » Controllo dell'accesso

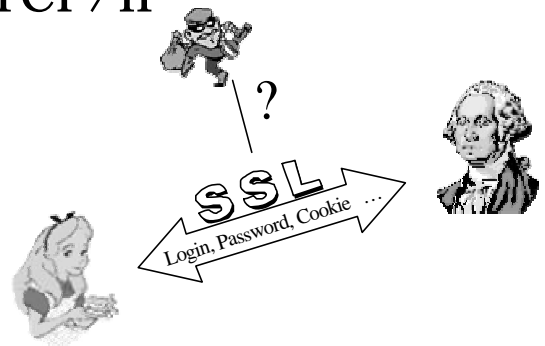
PARTE II: Transazioni sicure sul WEB

- » OpenSSL
- » Gestione di certificati digitali
- » ModSSL

TCP/IP



TCP/IP



Motivazioni

- TCP/IP consente di leggere ed alterare i dati che vengono inviati in rete
- Le informazioni scambiate tra 2 applicazioni su Internet passano per diverse organizzazioni
- Molti servizi richiedono il supporto delle stesse proprietà (autenticazione, integrità,...)

Il protocollo SSL

- SSL = Secure Socket Layer
- Socket = concetto di UNIX per network API
- Offre meccanismi di sicurezza alle applicazioni che usano il protocollo TCP
- E' uno standard per rendere sicuro il protocollo HTTP
- Altri protocolli usano SSL (NNTP, POP3, IMAP, ...)

Caratteristiche di SSL

- Fornisce l'autenticazione per le applicazioni server e client
- Cifra i dati prima di inviarli su un canale pubblico
- Garantisce l'integrità dell'informazione
- E' stato progettato per essere efficiente
- I principali algoritmi crittografici utilizzati vengono negoziati tra le parti

Uso diffuso di SSL

- Commercio elettronico
 - » Ordinazioni: le form con cui si ordina un prodotto vengono inviate usando SSL
 - » Pagamenti: quando viene inserito un numero di carta di credito, l'invio dei dati avviene usando SSL
- Accesso ad informazioni sicure
 - » La consultazione di informazioni accessibili solo da utenti "qualificati"
 - » L'invio di password o altri dati riservati

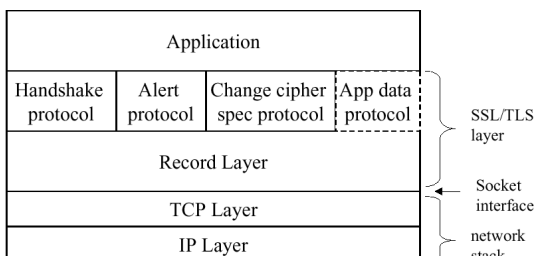
Storia di SSL

- Sviluppato ed introdotto da Netscape
- 1994 versione 1: diversi problemi, mai utilizzata
- 1994 versione 2: implementata in Navigator 1
- 1996 versione 3: implementata in Navigator 3
- 1999 TLS (Transport Layer Security) RFC 2246 evoluzione di SSL proposta dall'IETF

SSL: funzionalità

- Cifratura simmetrica
- Cifratura asimmetrica
- Firme digitali
- Certificati digitali (X509v.3)
- Specifiche chiare e formali
- Negoziazione dei parametri
- Handshake al momento della connessione
- Riutilizzo di parametri negoziati in precedenza

SSL e TCP/IP



Componenti di SSL

- Alert protocol
 - » Notifica situazioni anomale o segnala eventuali problemi
- Handshake protocol
 - » Permette alle parti di negoziare i diversi algoritmi necessari per la sicurezza delle transazioni
 - » Consente l'eventuale autenticazione tra le parti
- Change Cipher Spec protocol
 - » Impone l'esecuzione di un nuovo handshake per rinegoziare i parametri di sicurezza e ripetere l'autenticazione
- Record protocol
 - » Si occupa della compressione, del MAC e della cifratura

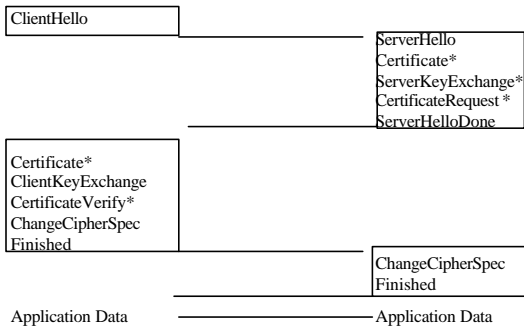
Ciphersuite di SSL

- Algoritmo per lo scambio di chiavi
- Algoritmo per l'autenticazione
- Algoritmo per la cifratura simmetrica
- Algoritmo per il message authentication code (MAC)
- Esempi
 - » EXP- RSA - RC4- MD5
 - ⇒ Kx= RSA(512), Au= RSA, Enc= RC4(40), Mac= MD5, exp
 - » RSA - DES- CBC3- SHA
 - ⇒ Kx= RSA, Au= RSA, Enc= 3DES(168), Mac= SHA1

Sessione sicura

- Una sessione sicura rappresenta una sequenza di valori che possono essere utilizzati con SSL
 - » valori segreti (calcolati durante l'handshake)
 - » ciphersuite (stabilita durante l'handshake)
- Stabilire tutti i parametri ogni volta che c'è una connessione può essere inefficiente. Una sessione può sopravvivere quindi tra più connessioni

Handshake di SSL



*messaggio opzionale

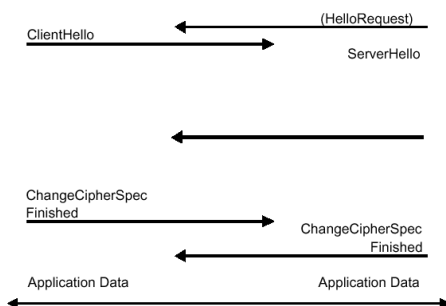
Costo di una sessione

- Client side
 - » Generazione di valori random
 - » Verificare il certificato digitale del server
 - » Generare dei valori random per la chiave
 - » Cifrare i valori random con la chiave pubblica del server
 - » Calcolare la chiave attraverso degli hash
- Server side
 - » Generazione di valori random
 - » Decifrare i valori inviati dal client
 - » Verificare il certificato del client e la firma di una challenge
 - » Calcolare la chiave attraverso degli hash



Un elevato rate di accessi mette in crisi il server

Handshake: Riesumazione



SSL: {Client, Server}Hello, ServerHelloDone

- Sono i primi messaggi inviati per stabilire i parametri di una sessione
- Permettono di scambiare valori Random generati da entrambe le parti
- Permettono alle parti di accordarsi su una ciphersuite
- Controllano la necessità di riesumare una sessione iniziata in precedenza
- Sono tutti obbligatori e l'unico senso di ServerHelloDone è comunicare che il blocco di messaggi del server è terminato

ClientHello

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<0..216-1>;
    CompressionMethod
        compression_methods<0..28-1>;
} ClientHello;
```

ServerHello

```
struct {
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
} ServerHello;
```

Autenticazione: Certificate, Certificate{Request, Verify}

- Sono i messaggi che consentono alle parti di autenticarsi
- Il messaggio Certificate contiene una lista di certificati
- Il certificato del server deve essere conforme con l'algoritmo di autenticazione specificato dalla ciphersuite concordata
- Il certificato client deve essere mandato solo se c'e' un messaggio CertificateRequest

Autenticazione: Certificate, Certificate{Request, Verify}

- L'eventuale certificato inviato dal client deve essere conforme e specifiche indicate nel messaggio CertificateRequest (può vincolare il tipo di certificato e le Certification Authority che lo hanno rilasciato)
- L'invio dei certificati non è obbligatorio ma quello server può essere necessario (dipende dalla ciphersuite stabilita)

KeyExchange

- Il server invia il messaggio ServerKeyExchange se il proprio certificato non è sufficiente per il tipo di autenticazione stabilito nella ciphersuite
- Il messaggio ClientKeyExchange è obbligatorio e con esso le parti hanno le informazioni necessarie per poter calcolare la chiave di cifratura simmetrica da utilizzare dopo l'handshake

ServerKeyExchange

```
struct {
    select (KeyExchangeAlgorithm) {
        case diffie_hellman:
            ServerDHParams params;
            Signature signed_params;
        case rsa:
            ServerRSAPParams params;
            Signature signed_params;
    };
} ServerKeyExchange;
```

ClientKeyExchange

```
struct {
    select (KeyExchangeAlgorithm) {
        case rsa: EncryptedPreMasterSecret;
        case diffie_hellman: ClientDiffieHellmanPublic;
    } exchange_keys;
} ClientKeyExchange;

struct {
    ProtocolVersion client_version;
    opaque random[48];
} PreMasterSecret;
```

CertificateVerify

- Il messaggio CertificateVerify viene inviato dal client solo se ha inviato il proprio certificato
- Contiene una firma digitale dell'hash dei messaggi scambiati fino a quel momento
- Il messaggio ServerKeyExchange oppure la decifrazione di ClientKeyExchange vengono usati con il messaggio Finished per l'identificazione del server

ChangeCipherSpec - Finished

- Con ChangeCipherSpec ogni parte indica all'altra che sta per usare gli algoritmi e le chiavi appena negoziati
- I messaggi Finished sono di testing e sono i primi messaggi che vengono inviati utilizzando gli ultimi algoritmi e chiavi concordati

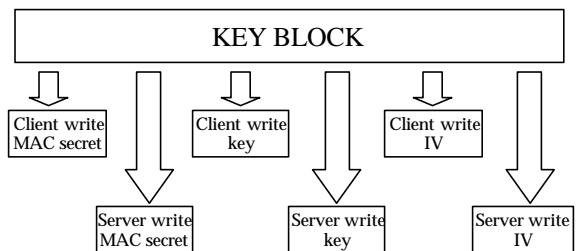
Calcolo delle chiavi

```
Master_secret =
    MD5(pre_master_secret +
        SHA('A' + pre_master_secret + ClientHello.random
            + ServerHello.Random)) +
    MD5(pre_master_secret +
        SHA('BB' + pre_master_secret + ClientHello.random
            + ServerHello.Random)) +
    MD5(pre_master_secret +
        SHA('CCC' + pre_master_secret +
            ClientHello.random + ServerHello.Random))
```

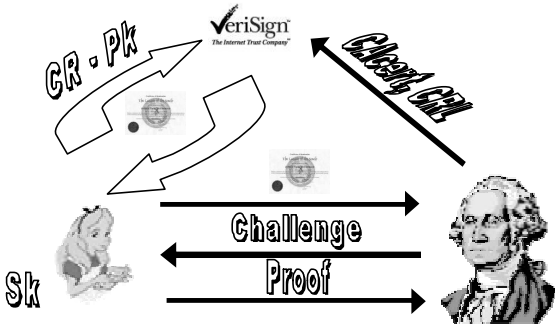
Calcolo delle chiavi

```
Key_block =
    MD5(master_secret +
        SHA('A' + master_secret + ClientHello.random +
            ServerHello.Random)) +
    MD5(pre_master_secret +
        SHA('BB' + master_secret + ClientHello.random
            + ServerHello.Random)) +
    MD5(master_secret +
        SHA('CCC' + master_secret +
            ClientHello.random + ServerHello.Random))
```

Calcolo delle chiavi



PKIX: Autenticazione



Challenge

```
enum { client(0x434C4E54), server(0x53525652) } Sender;

md5_hash = MD5(master_secret + pad2 +
MD5(handshake_messages +
Sender + master_secret + pad1));

sha_hash = SHA(master_secret + pad2 +
SHA(handshake_messages +
Sender + master_secret + pad1));
```

Firma in ClientVerify

```
CertificateVerify.signature.md5_hash
MD5(master_secret + pad_2 +
MD5(handshake_messages + master_secret + pad_1));
CertificateVerify.signature.sha_hash
SHA(master_secret + pad_2 +
SHA(handshake_messages + master_secret + pad_1));

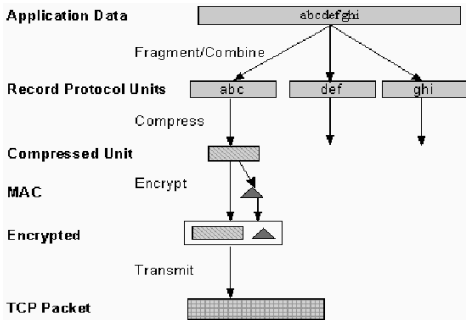
pad_1 0x36 ripetuto 48 volte per MD5 o 40 volte per SHA.
pad_2 0x5c ripetuto 48 volte per MD5 o 40 volte per SHA.
```

ChangeCipherSpec, Finished

```
struct {
enum { change_cipher_spec(1), (255) } type;
} ChangeCipherSpec;

struct {
opaque md5_hash[16];
opaque sha_hash[20];
} Finished;
```

Il compito di SSL (semplificato)



TLS: key_block

```
A(0) = seed
A(i) = HMAC_hash(secret, A(i-1))

P_hash(secret, seed) =
HMAC_hash(secret, A(1) + seed) +
HMAC_hash(secret, A(2) + seed) +
HMAC_hash(secret, A(3) + seed) + ...

PRF(secret, label, seed) =
P_MD5(S1, label + seed) XOR P_SHA -1(S2, label + seed);
dove secret = S1 + S2

key_block =
PRF(master_secret, "key expansion", server_random + client_random);
```

SSL: Analisi

- Connessione anonima
 - » server e client non presentano certificati
 - » scambi di chiavi con Diffie-Hellman
 - » attacco man in the middle
- Server autenticato
 - » con RSA autenticazione e scambi chiavi combinati
- Server e client autenticati
 - » entrambi provano di conoscere la chiave privata corrispondente alla pubblica presente nel certificato

SSL: Analisi

- 2001: viene dimostrato che calcolare il MAC e poi cifrare è meno sicuro rispetto a cifrare e poi calcolare il MAC
- 2002: viene presentato un attacco agli schemi di padding per i cifrari a blocchi in modalità CBC. L'attacco decifra blocchi e richiede:
 - » La disponibilità dei messaggi di errore
 - » Il proseguimento di una sessione in caso di erroreA causa del secondo vincolo è applicabile solo parzialmente ad SSL consentendo di indovinare l'ultimo byte con probabilità 254/255.

SSL: Analisi

- 2003: l'attacco agli schemi di padding per i cifrari a blocchi in modalità CBC viene esteso a SSL/TLS e richiede:
 - » Un'informazione ripetutamente cifrata (per esempio la password della posta elettronica)
 - » Un cifrario a blocchi in modalità CBC
 - » La possibilità di effettuare attacchi attivi tra client e server
 - » La possibilità di distinguere i tempi di esecuzione su errori differenti
- Tutti gli attacchi sono stati riparati nelle ultime release delle librerie che implementano SSL, ma chi ha aggiornato il proprio software ?

Utilizzo di SSL per il WEB

- Bisogna utilizzare un browser che supporti SSL
- Internet Explorer e Netscape Navigator supportano SSL
- E' possibile installare delle apposite patch per aggiungere alle vecchie versioni di tali browser il supporto di chiavi a 128 e 1024 bit
- E' possibile utilizzare un proxy col supporto di SSL

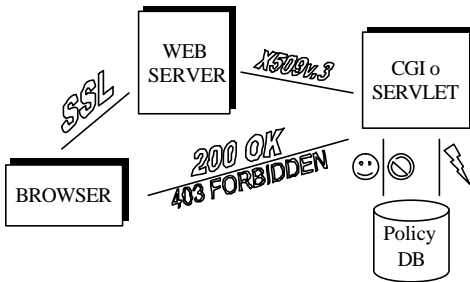
SSL: Access Control

- Autenticazione basata sull'indirizzo dell'host
 - » Soltanto alcuni indirizzi IP hanno l'accesso (spoofing)
- Autenticazione basata su username e password
 - » Tutto ciò che passa in rete è cifrato
 - » Viene evitato lo spoofing
 - » Ma i cookie sono in chiaro su disco
 - » Le password sono facilmente comunicate ad altri

SSL: Access Control

- Il server ha un database con i certificati degli utenti "qualificati" e le politiche di accesso
- Il web server richiede il certificato client durante l'handshake di SSL
- Il client invia il certificato richiesto dal server, di conseguenza, dopo aver stabilito la transazione sicura, potrà ottenere tutti i servizi consentiti da politiche di accesso in base all'identità evidenziata dal certificato digitale

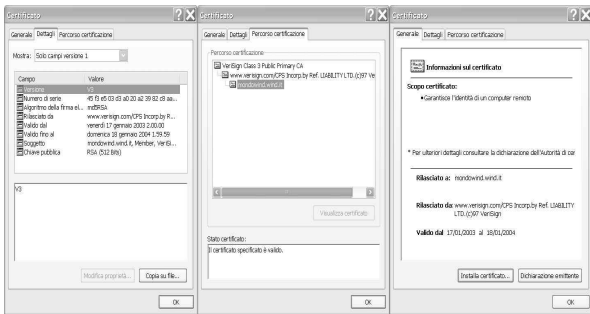
Access Control: Servlet o CGI



HTTPS: pagine WEB sicure



HTTPS: pagine WEB sicure



Disponibilità di SSL

- Implementazione di Netscape
 - » sslref
- Implementazione Open Source
 - » Openssl
- Supportato da Browser e Web Server
 - » https
- Applicazioni *SSL aware*
 - » pop3

OpenSSL

- OpenSSL è un package open source
- E' sottoposto a manutenzione continua
- E' utilizzato per lo sviluppo di importanti applicazioni (modSSL)
- Contiene implementazioni di vari algoritmi di crittografia
- Contiene implementazioni di Big Number, formati DER, PEM, pkcs7, pkcs12....
- Implementa il protocollo SSL/TLS
- Ha i comandi per gestire certificati digitali

OpenSSL: Certificati self-signed

- openssl req -config openssl.cnf -newkey rsa:512
 - days 1000 -nodes -keyout cakey.pem
 - out cacert.pem -x509 -new
- » req indica la richiesta di un nuovo certificato
- » x509 indica che il certificato deve essere self-signed
- » config indica il file con le configurazioni da usare per default
- » newkey specifica il formato della chiave
- » days indica la durata di validità
- » nodes indica che la chiave privata sia salvata in chiaro
- » keyout indica il nome del file con la chiave privata
- » out indica il nome del file col certificato

Certificato self-signed

```
C:\Ivan\Apa0pMod\openssl\bin>openssl req -new -x509 -out cacert.pem -keyout cakey.pem -days 365 -nodes -newkey RSA:1024 -config openssl.cnf
Using configuration from openssl.cnf
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'cakey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) []:IT
State or Province Name (full name) []:Salerno
Locality Name (eg. city) []:Fisciano
Organization Name (eg. company) []:Unisa
Organizational Unit Name (eg. section) []:Corso Security
Common Name (eg. your website domain name) []:CASecurity
Email Address []:

C:\Ivan\Apa0pMod\openssl\bin>_
```

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004 49

OpenSSL: richiesta di certificati

- openssl req -new -newkey rsa:512 -nodes -keyout Key.pem -out Req.pem -config openssl.cnf
 - » new indica che è una nuova richiesta di certificato
 - » config indica il file con le configurazioni da usare per default
 - » newkey specifica il formato della chiave
 - » nodes indica che la chiave privata sia salvata in chiaro
 - » keyout indica il nome del file con la chiave privata
 - » out indica il nome del file col certificato

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004 50

Richiesta di un certificato

```
C:\Ivan\Apa0pMod\openssl\bin>openssl req -new -out clientreq.pem -keyout clientkey.pem -newkey RSA:1024 -nodes -config openssl.cnf
Using configuration from openssl.cnf
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'clientkey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) []:IT
State or Province Name (full name) []:Salerno
Locality Name (eg. city) []:Fisciano
Organization Name (eg. company) []:Unisa
Organizational Unit Name (eg. section) []:Corso Security
Common Name (eg. your website domain name) []:Client
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:

C:\Ivan\Apa0pMod\openssl\bin>_
```

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004 51

OpenSSL: rilascio di certificati

- ```
openssl ca -policy policy_anything -out cert.pem -config openssl.cnf -infiles req.pem
```
- » config indica il file con le configurazioni da usare per default
  - » policy indica le politiche da utilizzare per il rilascio
  - » infiles indica il nome del file con la richiesta
  - » out indica il nome del file col certificato
  - » ca è l'opzione per la firma di un certificato

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004 52

## Rilascio di un certificato

```
C:\Ivan\Apa0pMod\openssl\bin>openssl ca -policy policy_anything -config openssl.cnf -out clientcert.pem -infiles clientreq.pem
Using configuration from openssl.cnf
Loading 'screen' into random state - done
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName :PRINTABLE:'IT'
stateOrProvinceName :PRINTABLE:'Salerno'
localityName :PRINTABLE:'Fisciano'
organizationName :PRINTABLE:'Unisa'
organizationalUnitName:PRINTABLE:'Corso Security'
commonName :PRINTABLE:'Client'
Certificate is to be certified until Jun 14 19:31:04 2004 GMT (365 days)
Sign the certificate? [y/n]y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

C:\Ivan\Apa0pMod\openssl\bin>_
```

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004 53

## L'utility x509 per la gestione dei certificati

- L'utility x509 di OpenSSL gestisce i certificati digitali
- Permette la conversione tra formati di certificati
- Consente la visualizzazione delle informazioni contenute in un certificato
- Permette di conoscere l'hash di un certificato da utilizzare per referenziarlo come certificato di un Certification Authority in una directory

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004 54

# L'utility x509

- Ecco le principali opzioni dell'utility:
  - » -in indica il file di input col certificato
  - » -out indica il file di output col certificato
  - » -inform indica il formato di input
  - » -outform indica il formato di output
  - » -text visualizza le informazioni contenute nel certificato
  - » -noout non visualizza il certificato nel suo formato
  - » -hash visualizza l'hash del certificato nel formato necessario per usarlo come una CA in una directory

# Uso di X509

```
C:\Ivan\ApaOpMod\openssl\bin>openssl x509 -in clientcert.pem -text -noout
Certificate
Data:
 Version: 3 (0x2)
 Serial Number: 3 (0x3)
 Signature Algorithm: md5WithRSAEncryption
 Issuer: C=IT, ST=Salerno, L=Fisciano, O=Unisa, OU=Corso Sicurezza, CN=Aut
 orita del corso di sicurezza
 Validity
 Not Before: Jun 15 19:31:04 2003 GMT
 Not After: Jun 14 19:31:04 2004 GMT
 Subject: C=IT, ST=Salerno, L=Fisciano, O=Unisa, OU=Corso Security, CN=C
 ient
 Subject Public Key Info:
 Public Key Algorithm: rsaEncryption
 rsa Public Key: (1024 bit)
 Modulus (1024 bit):
 00:e1:35:61:66:78:9a:63:6a:a2:52:3c:df:b4:9b:
 fe:60:b7:c9:4f:db:8c:02:d6:de:57:1d:7b:b6:72:
 67:00:72:fe:d4:19:a5:40:20:58:59:cd:0e:f9:de:
 0f:de:ed:cf:ff:38:a3:8f:6d:67:32:75:12:0e:67:
 51:7e:12:78:5b:13:9a:89:d7:98:8e:de:09:66:f0:
 87:2e:d5:7e:55:2a:13:da:72:0c:01:50:e2:2f:2c:
 00:74:5f:fe:8f:de:c5:10:41:ee:bc:52:b4:bc:3d:
 c0:81:dd:89:97:5b:c7:47:3b:60:92:29:6f:22:03:
 b4:2f:52:84:d1:ab:60:16:cd
 Exponent: 65537 (0x10001)
 Signature Algorithm: md5WithRSAEncryption
 25:c8:7f:c4:4e:fa:50:05:c1:c9:4f:42:9b:cd:91:f6:9d:d2:
 e9:7d:03:a8:85:aa:44:05:33:4f:ca:05:fc:24:ff:05:4a:ae:
 5b:df:b1:9a:f0:ea:9b:ff:12:47:a4:d3:2c:9e:63:11:32:56:
 69:81:e5:23:0a:3a:50:84:67:21:cb:85:1c:ed:f5:12:cc:f2:
 3f:bd:ef:aa:0a:70:bc:8a:f2:ca:02:d3:d6:9c:a1:32:6b:db:
 b0:81:01:30:7b:e2:b0:14:e5:08:02:7b:08:6b:8e:a0:c0:5d:
```

# OpenSSL – Conversione in p12

- Il formato PKCS12 viene utilizzato per importare certificati e chiavi in un browser

```
openssl pkcs12 -export -chain -CAfile cacert.pem
-inkey Key.pem -name Abc -in Cert.pem -out
Cert.p12
```

- Vengono indicati i file con le informazioni necessarie per ottenere un file in formato PKCS12

# Uso di PKCS12

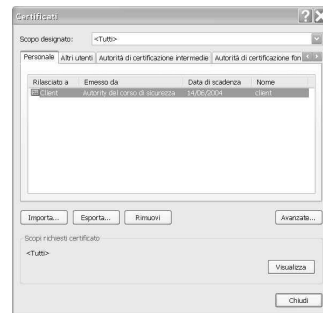
```
C:\Ivan\ApaOpMod\openssl\bin>openssl pkcs12 -export -inkey clientkey.pem -name
client -in clientcert.pem -out clientcert.p12
Loading 'screen' into random state - done
Enter Export Password:
Verifying password - Enter Export Password:

C:\Ivan\ApaOpMod\openssl\bin>_
```

# Importazione in IE



# Importazione in IE



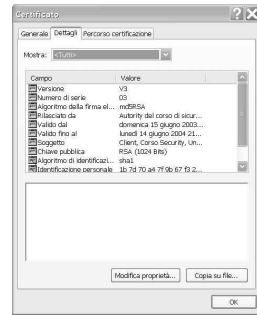
## Importazione in IE



Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004

61

## Importazione in IE



Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004

62

## Importazione in IE



Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004

63

## SSLClient con OpenSSL

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <openssl/ssl.h>
#define LEN 1024
#define PORT 3000

main(int argc, char **argv){

 char CAfile[]="CACert.pem";
 char buff_out[]="Prova di Invio Sicuro";
 char buff_in[LEN];
 SSL *ssl;
 SSL_CTX *ctx;
```

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004

64

## SSLClient con OpenSSL

```
int sd, cnt;
struct sockaddr_in servaddr;
if (argc != 2) {
 fprintf(stderr, "usage: %s <IPaddress>\n", argv[0]);
 exit(1);
}
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
 perror("opening socket");
 exit(1);
}
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(PORT);
if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0) {
 perror("inet_pton error");
 exit(1);
}
```

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004

65

## SSLClient con OpenSSL

```
SSLLeay_add_ssl_algorithms(); //initialize the supported algorithms
ctx = SSL_CTX_new(SSLv3_client_method()); // create a secure context
SSL_CTX_load_verify_locations(ctx, CAfile, NULL);
ssl = SSL_new(ctx); // create a free and secure connetion

SSL_set_fd(ssl, sd); // assign a file descriptor

if (connect(sd, (struct sockaddr *) &servaddr, sizeof(servaddr)) < 0){
 perror("connect error");
 exit(1);
}

SSL_set_verify(ssl, SSL_VERIFY_PEER, NULL);
if (SSL_connect(ssl) < 0){
 fprintf(stderr, "Errore in SSL_connect\n");
 exit(1);
}
```

Ivan Visconti SSL, TLS ed OpenSSL 4 giugno 2004

66

## SSLClient con OpenSSL

```
// do a secure and private connect
SSL_write(ssl, buff_out, strlen(buff_out)); // do a secure write
SSL_read(ssl, buff_in, LEN); // do a secure read
printf("Ho ricevuto:\n\t%s\n", buff_in);
SSL_shutdown(ssl); // close a secure connection
SSL_free(ssl); // free memory
SSL_CTX_free(ctx); // free memory
}
```

## SSLServer con OpenSSL

```
#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <openssl/ssl.h>
#define LEN 1024
#define PORT 3000

main(int argc, char **argv){
 char filename[]="CertServer.pem";
 char secretkey[]="KeyServer.pem";
 char buff[LEN];
 SSL *ssl;
 SSL_CTX *ctx;
 int sd, connsd, cnt, lencliaddr;
 struct sockaddr_in servaddr, clientaddr;
```

## SSLServer con OpenSSL

```
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) <0){
 perror("opening socket");
 exit(1);
}

bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

if (bind(sd, (struct sockaddr *) &servaddr, sizeof(servaddr)) < 0){
 perror("Error in binding");
 exit(1);
}

listen(sd, 5);
```

## SSLServer con OpenSSL

```
SSL_load_ssl_algorithms(); // initialize the supported algorithms

ctx = SSL_CTX_new(SSLv3_server_method()); // create a secure context

// certificate to be used
if(!SSL_CTX_use_certificate_file(ctx, filename, SSL_FILETYPE_PEM)){
 fprintf(stderr, "Non trovo certificato in %s\n", filename);
 exit(1);
}

// private key of the certificate
if(!SSL_CTX_use_PrivateKey_file(ctx, secretkey, SSL_FILETYPE_PEM)){
 fprintf(stderr, "Non trovo chiave in %s\n", filename);
 exit(1);
}
```

## SSLServer con OpenSSL

```
for (;;) {
 ssl = SSL_new(ctx); // create a free and secure connection
 connsd = accept(sd, (struct sockaddr *)&clientaddr, (void *)&lencliaddr);
 SSL_set_fd(ssl, connsd); // assign a file descriptor
 // do a secure accept
 if(SSL_accept(ssl)<0){
 fprintf(stderr, "Errore in SSL Accept\n");
 exit(1);
 }
 cnt = SSL_read(ssl, buff, LEN); // do a secure read
 buff[cnt] = 0;
 SSL_write(ssl, buff, cnt+1);
 SSL_shutdown(ssl); // close a secure connection
 SSL_free(ssl);
}
SSL_CTX_free(ctx);
}
```

## SSLClient con OpenSSL: autenticazione client

```
SSL_CTX_load_verify_locations(ctx, CAfile, NULL);
// certificate to be used
if(!SSL_CTX_use_certificate_file(ctx, filename, SSL_FILETYPE_PEM)){
 fprintf(stderr, "Non trovo certificato in %s\n", filename);
 exit(1);
}

// private key of the certificate
if(!SSL_CTX_use_PrivateKey_file(ctx, secretkey, SSL_FILETYPE_PEM)){
 fprintf(stderr, "Non trovo chiave in %s\n", filename);
 exit(1);
}
```

## SSLServer con OpenSSL: autenticazione client

```
SSL_CTX_load_verify_locations(ctx, CAfile, NULL);
SSL_CTX_set_verify(ctx,
SSL_VERIFY_PEER|SSL_VERIFY_FAIL_IF_NO_PEER_CERT, NULL);
```

```
.....
.....
```

```
cert=SSL_get_peer_certificate(ssl);
X509_NAME_oneline(X509_get_issuer_name(cert), buff, LEN);
fprintf(stderr, "Issuer: %s\n", buff);
X509_NAME_oneline(X509_get_subject_name(cert), buff, LEN);
fprintf(stderr, "Subject: %s\n", buff);
name = strstr(buff, "/CN=")+4;
fprintf(stderr, "Common Name: %s\n", name);
if (strcmp(name, "Client")!=0) {
 fprintf(stderr, "Accesso non autorizzato\n");
 exit(1);
}
```

Ivan Visconti

SSL, TLS ed OpenSSL

4 giugno 2004

73

## OpenSSL: s\_server e s\_client

- Le utility s\_server e s\_client vengono distribuite con OpenSSL e sono uno dei principali strumenti di debug utilizzati da chi sviluppa applicazioni client/server sicure
- Possono essere eseguite "indipendentemente" l'una dall'altra e sono configurabili con sequenza di argomenti che consentono di scegliere il tipo di connessione SSL desiderata

Ivan Visconti

SSL, TLS ed OpenSSL

4 giugno 2004

74

## L'utility s\_server

- L'utility s\_server è parte del package OpenSSL
- E' un server SSL utile per il debug di applicazioni client col supporto di SSL
- E' possibile configurare l'esecuzione di questa utility impostando degli argomenti nella riga di comando
- Prevede ad esempio l'eventuale uso di certificati, autenticazione client, selezione di cipher suite, della versione del protocollo

Ivan Visconti

SSL, TLS ed OpenSSL

4 giugno 2004

75

## L'utility s\_server: parametri per l'esecuzione

|              |                                                            |
|--------------|------------------------------------------------------------|
| -accept arg  | porta TCP/IP del server (default 4433)                     |
| -verify arg  | richiede l'autenticazione client                           |
| -Verify arg  | fallisce la connessione se non c'è autenticazione client   |
| -cert arg    | indica il file col certificato server (default server.pem) |
| -key arg     | indica il file con la chiave privata (default server.pem)  |
| -dcert arg   | eventuale secondo certificato (in generale DSA)            |
| -dkey arg    | eventuale seconda chiave (in generale DSA)                 |
| -dhparam arg | file con i parametri DH                                    |
| -nbio        | l'esecuzione avviene con socket non bloccante              |
| -debug       | vengono visualizzate maggiori informazioni per il debug    |
| -bugs        | l'esecuzione tollera alcuni noti bug                       |

Ivan Visconti

SSL, TLS ed OpenSSL

4 giugno 2004

76

## L'utility s\_server: parametri per l'esecuzione

|             |                                                       |
|-------------|-------------------------------------------------------|
| -CApath arg | directory con i certificati delle CA                  |
| -CAfile arg | file con i certificati delle CA                       |
| -nocert     | i certificati non vengono utilizzati (Anon-DH)        |
| -cipher arg | uso di particolari cipher suite                       |
| -ssl2       | uso di SSLv2                                          |
| -ssl3       | uso di SSLv3                                          |
| -tls1       | uso di TLSv1                                          |
| -no_ssl2    | non uso di SSLv2                                      |
| -no_ssl3    | non uso di SSLv3                                      |
| -no_tls1    | non uso TLSv1                                         |
| -no_dhe     | non uso di ephemeral DH                               |
| -www        | Risposta a GET / con una pagina di prova              |
| -WWW        | Risposta a 'GET /<path> HTTP/1.0' con il file /<path> |

Ivan Visconti

SSL, TLS ed OpenSSL

4 giugno 2004

77

## L'utility s\_client: parametri per l'esecuzione

|                            |                                                      |
|----------------------------|------------------------------------------------------|
| -connect hostport          | indica il server desiderato (default localhost:4433) |
| -verify arg                | imposta la verifica del certificato del server       |
| -cert arg                  | indica il certificato da usare                       |
| -key arg                   | indica la chiave da usare                            |
| -CApath arg                | indica la directory con i certificati delle CA       |
| -CAfile arg                | indica il file con i certificati delle CA            |
| -reconnect                 | interrompe la connessione e la riprende (riesuma)    |
| -showcerts                 | mostra i certificati ricevuti                        |
| -debug                     | visualizza maggiori informazioni                     |
| -nbio                      | usa socket non bloccanti                             |
| -ssl2/ssl3/tls1            | imposta un solo protocollo                           |
| -no_tls1/-no_ssl3/-no_ssl2 | disabilita qualche protocollo                        |
| -bugs                      | Imposta l'uso di soluzioni ai bug comuni             |
| -cipher                    | specifica le cipher suite                            |

Ivan Visconti

SSL, TLS ed OpenSSL

4 giugno 2004

78

# Una piattaforma di sviluppo

- Server web:
  - » apache
- Implementazione di SSL
  - » openssl
- Integrazione server web + ssl
  - » modssl

# Apache

- è un web server potente e flessibile
- implementa il protocollo HTTP/1.1
- è configurabile ed estendibile con moduli esterni
- i sorgenti sono utilizzabili liberamente
- è disponibile per Windows NT/9x, Netware 5.x, OS/2, e molte versioni di Unix
- è sottoposto a continua manutenzione
- è usato dal 56% dei web server su Internet
- è ampiamente documentato: <http://httpd.apache.org>

# ModSSL

Sviluppato da Ralf Engelschall che lo definisce così:

“mod\_ssl combines the flexibility of Apache with the security of OpenSSL”

è ampiamente documentato: [www.modssl.org](http://www.modssl.org)

# Autenticazione con ModSSL

## FakeBasicAuth

- » Consente di concedere l'accesso in base al distinguished name (DN) dell'utente
- » Il DN viene usato come nome utente e la password è fissa
- » Non si possono effettuare operazioni con la CA
- » Non è possibile raggruppare utenti che hanno qualcosa in comune

# Autenticazione con ModSSL

## SSLRequire

- » Viene utilizzata un'espressione regolare
- » E' possibile utilizzare le variabili di environment che usano i CGI
- » Incorpora i vantaggi dell'autenticazione host based
- » Può forzare una rinegoziazione per i seguenti motivi
  - ◆ Cipher insufficiente
  - ◆ Richiesta del certificato client
  - ◆ Richiesta di un diverso certificato client

# SSLRequire

In questo esempio si richiede l'accesso per un certificato con dei vincoli sulla data di rilascio e sul DN

```
SSLRequire (
 %{ SSL_CLIENT_S_DN_O } eq "Snake Oil, Ltd." \
 and %{ SSL_CLIENT_S_DN_OU } in { "staff", "CA", "Dev" } \
 and %{ SSL_CLIENT_V_START } >= 20000504 \
)
```

## Autenticazione con CGI

- ModSSL esporta nelle variabili di environment dei CGI diverse informazioni
- E' possibile da un CGI accedere ai campi del certificato client
- E' possibile da un CGI accedere e informazioni sulla cipher suite

## ModSSL: direttive di configurazione

- SSLPassPhraseDialog
  - » Specifica la chiave con cui è cifrata la chiave privata del server
- SSLRandomSeed
  - » Consente di impostare una base per la generazione di valori random
- SSLSessionCache
  - » Consente di specificare l'uso di una cache
- SSLEngine
  - » Imposta l'uso di SSL

## ModSSL: Direttive di configurazione

- SSLProtocol
  - » Indica la versione del protocollo da utilizzare
- SSLCipherSuite
  - » Indica le ciphersuite desiderate
- SSLCertificateFile
  - » Indica il file col certificato del server
- SSLCertificateKeyFile
  - » Indica il file con la chiave privata
- SSLCACertificateFile
  - » Indica il file con il certificato della CA

## ModSSL: direttive di configurazione

- SSLVerifyClient
  - » Imposta la richiesta del certificato del client
- SSLLog
  - » Indica il file di log
- SSLOptions
  - » Configura alcune opzioni tra cui la possibilità di esportare informazioni ai CGI in variabili di environment
- SSLRequireSSL
  - » Nega l'accesso quando non è in uso SSL

## Alias per le CipherSuite

### Key Exchange Algorithm:

kRSA RSA key exchange  
kDhR Diffie-Hellman key exchange with RSA key  
kDhD Diffie-Hellman key exchange with DSA key  
kEDH Ephemeral (temp.key) Diffie-Hellman key exchange

### Authentication Algorithm:

aNULL No authentication  
aRSA RSA authentication  
aDSS DSS authentication  
aDH Diffie-Hellman authentication

## Alias per le CipherSuite

### Cipher Encoding Algorithm:

|       |                     |
|-------|---------------------|
| eNULL | No encoding         |
| DES   | DES encoding        |
| 3DES  | Triple-DES encoding |
| RC4   | RC4 encoding        |
| RC2   | RC2 encoding        |
| IDEA  | IDEA encoding       |

## Alias per le CipherSuite

MAC Digest Algorithm:

|      |                     |
|------|---------------------|
| MD5  | MD5 hash function   |
| SHA1 | SHA-1 hash function |
| SHA  | SHA hash function   |

## Alias per le CipherSuite

Alias:

|          |                                                     |
|----------|-----------------------------------------------------|
| SSLv2    | SSL version 2.0 ciphers                             |
| SSLv3    | SSL version 3.0 ciphers                             |
| TLSv1    | TLS version 1.0 ciphers                             |
| EXP      | export ciphers                                      |
| EXPORT40 | 40-bit export ciphers only                          |
| EXPORT56 | 56-bit export ciphers only                          |
| LOW      | low strength ciphers (no export, single DES)        |
| MEDIUM   | ciphers with 128 bit encryption                     |
| HIGH     | ciphers using Triple-DES                            |
| RSA      | ciphers using RSA key exchange                      |
| DH       | ciphers using Diffie-Hellman key exchange           |
| EDH      | ciphers using Ephemeral Diffie-Hellman key exchange |
| ADH      | ciphers using Anonymous Diffie-Hellman key exchange |
| DSS      | ciphers using DSS authentication                    |
| NULL     | ciphers using no encryption                         |

## Configurazione del web server

(Da inserire in httpd.conf)

```
Port 80
Listen 80
Listen 443
LoadModule ssl_module modules/mod_ssl.so
AddModule mod_ssl.c
<virtualhost 127.0.0.1:443>
 DocumentRoot C:/...
 SSLEngine on
 SSLCertificateFile servercert.pem
```

## Configurazione del web server

```
SSLCertificateKeyFile serverkey.pem
SSLCACertificateFile CAacerts.pem
SSLCipherSuite RSA
SSLLog logs/ssl_engine_log
<Location /cgi-bin/>
 SSLVerifyClient require
 SSLOptions +StdEnvVars
 SSLOptions +ExportCertData
</Location>
</VirtualHost>
```