Protocolli SSL e TLS con OpenSSL

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@dia.unisa.it

http://www.dia.unisa.it/professori/ads

Maggio 2017

OpenSSL s_client ed s_server

- I comandi **s_server** ed **s_client** sono i principali strumenti forniti da OpenSSL per il debug di applicazioni client/server che utilizzano SSL/TLS
  - Possono essere eseguiti "indipendentemente" l'uno dall'altro
  - Sono configurabili mediante opportuni parametri, che permettono di scegliere il tipo di connessione SSL/TLS che si intende stabilire

OpenSSL s_client

- Il comando **s_client**
  - Permette di realizzare tutte le funzioni di un semplice client SSL/TLS, che può essere utilizzato per connettersi ad un server che supporta tali protocolli
  - Fornisce funzionalità molto simili a quelle offerte da Telnet
    - Ma non supporta tale protocollo
  - Il comando è utile soprattutto come strumento diagnostico per la creazione e la configurazione di server SSL/TLS

OpenSSL s_client

Struttura generale del comando s_client

```
openssl s_client args
```

- **args**
  - **-connect host:port** Server e porta a cui connettersi (default localhost:4433)
  - **-CApath arg** Directory con i certificati delle CA
  - **-CAfile arg** File con i certificati delle CA
  - **-debug** Visualizza ulteriori informazioni per il debug
  - **-cipher** Specifica le chipersuite
  - **-verify arg** Imposta la verifica del certificato del server
  - **-cert arg** Certificato da usare, in formato PEM
  - **-key arg** Chiave privata relativa al certificato usato
  - **-msg** Mostra i messaggi del protocollo
  - **-showcerts** Mostra tutti i certificati presenti nella certificate chain
  - **-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1,…** Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
  - **-starttls prot** Permette di specificare quale protocollo si intende utilizzare over SSL/TLS. Attualmente, i protocolli supportati sono: **smtp**, **pop3**, **imap**, **ftp**, **xmpp**

OpenSSL s_client

Struttura generale del comando s_client

```
openssl s_client args
```

- **args**
  - **-connect host:port** Se… porta a cui connettersi (default localhost:4433)
  - **-CApath arg** Directo…
  - **-CAfile arg** File con…
  - **-debug** Visualizza ulte…
  - **-cipher** Specifica le cipher suite
  - **-verify arg** Imposta la verifica del certificato del server
  - **-cert arg** Certificato da usare, in formato PEM
  - **-key arg** Chiave privata relativa al certificato usato
  - **-msg** Mostra i messaggi del protocollo
  - **-showcerts** Mostra tutti i certificati presenti nella certificate chain
  - **-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1,…** Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
  - **-starttls prot** Permette di specificare quale protocollo si intende utilizzare over SSL/TLS. Attualmente, i protocolli supportati sono: **smtp**, **pop3**, **imap**, **ftp**, **xmpp**

Per ottenere la lista completa delle opzioni del comando **s_client** è possibile utilizzare **man s_client**

OpenSSL s_client

Mediante il comando **s_client** è possibile interrogare un server che offre connessioni SSL/TLS per uno specifico protocollo. Nel seguente esempio, interrogheremo un server SMTP, sulla porta 587

```
openssl s_client -msg -connect smtp.unisa.it:587 -starttls smtp
```

Handshake

```
CONNECTED(00000003)
>>> SSL 2.0 [length 0080], CLIENT-HELLO
<<< TLS 1.0 Handshake [length 0051], ServerHello
<<< TLS 1.0 Handshake [length 0a3e], Certificate
<<< TLS 1.0 Handshake [length 028c], ServerKeyExchange
<<< TLS 1.0 Handshake [length 0074], CertificateRequest
<<< TLS 1.0 Handshake [length 0004], ServerHelloDone
>>> TLS 1.0 Handshake [length 0007], Certificate
>>> TLS 1.0 Handshake [length 0086], ClientKeyExchange
>>> TLS 1.0 ChangeCipherSpec [length 0001]
>>> TLS 1.0 Handshake [length 0010], Finished
<<< TLS 1.0 ChangeCipherSpec [length 0001]
<<< TLS 1.0 Handshake [length 0010], Finished
---
```

OpenSSL s_client

Mediante il comando **s_client** è possibile interrogare un server che offre connessioni SSL/TLS per uno specifico protocollo. Nel seguente esempio, interrogheremo un server SMTP, sulla porta 587

```
openssl s_client -msg -connect smtp.unisa.it:587 -starttls
smtp
```

Certificate Chain

```
---
Certificate chain
 0 s:/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
---
```

OpenSSL s_client

Mediante il
connessioni
interrogher

```
openss
smtp
```

- Per ciascun certificato, la prima riga mostra il *subject* e la seconda l'*issuer*
- Il primo certificato della certificate chain è il certificato foglia (certificato 0)
- Si prosegue scorrendo la certificate chain verso il basso, verificando che l'issuer del certificato corrente coincida col subject del prossimo certificato
- L'ultimo issuer presente nella certificate chain può puntare a qualche certificato root che non è presente nella chain
  - Oppure, se si tratta di un certificato self-signed, può puntare a se stesso

Certificate Chain

```
---
Certificate chain
 0 s:/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
---
```

OpenSSL s_client

Mediante il comando **s_client** è possibile interrogare un server che offre connessioni SSL/TLS per uno specifico protocollo. Nel seguente esempio, interrogheremo un server SMTP, sulla porta 587

```
openssl s_client -msg -connect smtp.unisa.it:587 -starttls
smtp
```

Certificate Chain

```
---
Certificate chain
 0 s:/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
---
```

Certificato 1

OpenSSL s_client

Mediante il comando **s_client** è possibile interrogare un server che offre connessioni SSL/TLS per uno specifico protocollo. Nel seguente esempio, interrogheremo un server SMTP, sulla porta 587

```
openssl s_client -msg -connect smtp.unisa.it:587 -starttls
smtp
```

Certificate Chain

```
---
Certificate chain
 0 s:/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
---
```

Certificato 2

OpenSSL s_client

Mediante il comando **s_client** è possibile interrogare un server che offre connessioni SSL/TLS per uno specifico protocollo. Nel seguente esempio, interrogheremo un server SMTP, sulla porta 587

```
openssl s_client -msg -connect smtp.unisa.it:587 -starttls
smtp
```

Certificate Chain

```
---
Certificate chain
 0 s:/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
---
```

Subject

OpenSSL s_client

Mediante il comando **s_client** è possibile interrogare un server che offre connessioni SSL/TLS per uno specifico protocollo. Nel seguente esempio, interrogheremo un server SMTP, sulla porta 587

```
openssl s_client -msg -connect smtp.unisa.it:587 -starttls
smtp
```

Certificate Chain

```
---
Certificate chain
 0 s:/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
---
```

Issuer

OpenSSL s_client

```
Server certificate
-----BEGIN CERTIFICATE-----
MIIFLjCCBBagAwIBAgIQBo01MCrIE+SY6azANBgkqhkiG9w0BAQsFADBk
MQswCQYDVQQGEwJOTDEWMBQGA                                  UEBxMJ
QW1zdGVyZGFtMQ8wDQYDVQQKE                                  BTU0wg
Q0EgMzAeFw0xNjAxMTMwMDAwM                                  NVBAYT
AklUMREwDwYDVQQIEwhDYW1wYW5pYTERMA8GA1UEBxMIRmlzY2lhbm8xKzApBgNV
BAoMIlVuaXZlcnNpdMOgIGRlZ2xpIFN0dWRpIGRpIFNhbGVybm8xFjAUBgNVBAMT
DXNtdHAudW5pc2EuaXQwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDU
rg4k8EfLoho/z5seVoSfFWJbJUQtSNiat+q4509eif2XAH7mGCKWouoc4wGnUWq+
7wDqSyQHqBhcRmXh5hX6Zz67YeEjq1B4bILZdTNCs+utYX3Wuzp9cWIDJnq9Dp66
mUfceYv9mfaNuy8/RJg/pUPkNEgth76UKuO8Qxjy3im1HaQVq22qX14jznpg5ZBI
IR89uh3EAVfUkb50lfCCz2vrI8zgxNCto4wC0oGmO7epdKzEzGm8ZhR5wnayxFOe
hk1PyGDN+ngNWibcogI103DhwmFpizedfwx0CPJwOdrB/mEAzaqNaw1kOrK104Nu
f2Z+Rv3m6B9spReFkr0TAgMBAAGjggHGMIIBwjAfBgNVHSMEGDAWgBRn/YggFCeY
xwnSJRm76VERY3VQYjAdBgNVHQ4EFgQUllzCPjZTiDzPW4V+6zhr7tu37KAwGAYD
VR0RBBEwD4INc210cC51bmlzYS5pdDAOBgNVHQ8BAf8EBAMCBaAwHQYDVR0lBBYw
FAYIKwYBBQUHAwEGCCsGAQUFBwMCMGsGA1UdHwRkMGIwL6AtoCuGKWh0dHA6Ly9j
cmwzLmRpZ2ljZXJ0LmNvbS9URVJFTkFTU0xDQTMuY3JsMC+gLaArhilodHRwOi8v
Y3JsNC5kaWdpY2VydC5jb20vVEVSRU5BU1NMQ0EzLmNybDBMBgNVHSAERTBDMDcG
CWCGSAGG/WwBATAqMCgGCCsGAQUFBwIBFhxodHRwczovL3d3dy5kaWdpY2VydC5j
b20vQ1BTMAgGBmeBDAECAjBuBggrBgEFBQcBAQRiMGAwJAYIKwYBBQUHMAGGGGh0
dHA6Ly9vY3NwLmRpZ2ljZXJ0LmNvbTA4BggrBgEFBQcwAoYsaHR0cDovL2NhY2Vy
dHMuZGlnaWNlcnQuY29tL1RFUkVOQVNTTENBMy5jcnQwDAYDVR0TAQH/BAIwADAN
BgkqhkiG9w0BAQsFAAOCAQEAMhSymPW8ip8pv1Lice0xWEafOo7z+Urpz8wf3Cqh
/+zhIZs98KcsKMQ2UbfwnQMvC9NL                                  eoZ
nYm9gRzuGviIH3UTA7Jxst42q4oP                                  hUv
7Xv4gLZwTSERmGvQAH61S3a8RZNw                                  CE/
Z+cDLo9p4e57m0h4Ch6JRb7+vIOoe                                  CNM
BkDgt8hMBJuMGRz8IXoM2TQ7uVR0                                  2VzkV0ItHyQ==
-----END CERTIFICATE-----
subject=/C=IT/ST=Campania/L=Fisciano/O=Universit\xC3\xA0 degli Studi di Salerno/CN=smtp.unisa.it
issuer=/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
---
```

Certificato del server, codificato in Base64

Subject ed issuer del certificato server

OpenSSL s_client

```
---
Acceptable client certificate CA names
/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
---
SSL handshake has read 3891 bytes and written 375 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: N
Expansion: NO
SSL-Session
    Key-Arg   : None
    Start Time: 1488648232
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
```

Versione utilizzata del protocollo TLS/SSL e ciphersuite supportata

Elenco delle CA accettate dal server come issuer per i certificati dei client. Il client può utilizzare questo elenco per scegliere il certificato appropriato

OpenSSL s_client

```
---
Acceptable client certificate CA names
/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
---
SSL handshake has read 3891 bytes and written 375 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: E28EC8BCEFA000840E91ECE286549
    Session-ID-ctx:
    Master-Key: 86EAA03A328CD14AF19774602A2B9                                    0DA4973300D9
    Key-Arg   : None
    Start Time: 1488648232
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
```

Informazioni relative alla chiave pubblica del server, alla rinegoziazione dei parametri, alla compressione ed all'espansione

OpenSSL s_client

```
---
Acceptable client certificate CA names
/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
---
SSL handshake has read 3891 bytes and written 375 bytes
---
New, TLSv1/SSLv3, Cipher is DHE-RSA-AES256-SHA
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID: E28EC8BCEFA000840E91ECE2865494D605758632ED2A45D45DAE201D70950CE8
    Session-ID-ctx:
    Master-Key: 86EAA03A328CD14AF19774602A2B92D4021EF6EAEDAE9B65222EB95E55FF4EF803F36772C219AB2DE9E50DA4973300D9
    Key-Arg   : None
    Start Time: 1488648232
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
```

Informazioni relative alla sessione SSL/TLS corrente

OpenSSL s_server

- Il comando **s_server**
  - Permette di realizzare tutte le funzioni di un server SSL/TLS
  - Risulta utile per il debug di applicazioni client che supportano SSL/TLS
  - È possibile configurare l'esecuzione di questo comando attraverso l'impostazione di opportuni parametri
    - Ad es., eventuale uso di certificati, autenticazione client, selezione della ciphersuite, versione del protocollo, etc.

OpenSSL s_server

Struttura generale del comando s_server

```
openssl s_server args
```

- **args**
  - **-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** File col certificato server
  - **-key arg** File con la chiave privata
  - **-debug** Visualizza ulteriori informazioni per il debug
  - **-CApath arg** Directory con i certificati delle *CA*
  - **-CAfile arg** File con i certificati delle *CA*
  - **-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1,…** Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
  - **-cipher arg** Specifica le chipersuite
  - **-www** Risposta a GET / con una pagina di prova

OpenSSL s_server

Struttura generale del comando s_server

```
openssl s_server args
```

- **args**
  - **-accept arg** Porta TCP/ server (default 4433)
  - **-verify arg** Richiedo
  - **-Verify arg** Fallisce icazione client
  - **-cert arg** File col ce
  - **-key arg** File con la chiave privata
  - **-debug** Visualizza ulteriori informazioni per il debug
  - **-CApath arg** Directory con i certificati delle CA
  - **-CAfile arg** File con i certificati delle CA
  - **-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1,…** Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
  - **-cipher arg** Specifica le chipersuite
  - **-www** Risposta a GET / con una pagina di prova

Per ottenere la lista completa delle opzioni del comando **s_server** è possibile utilizzare **man s_server**

OpenSSL s_server

Mediante il comando **s_server** è possibile creare un server che offre connessioni SSL/TLS. Nel seguente esempio, creeremo un server in ascolto sulla porta 12345

```
openssl s_server -accept 12345 -cert server-cert.pem -key
private/server-key.pem
```

```
Enter pass phrase for private/server-key.pem:
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
```

Il server è stato avviato ed è pronto ad accettare richieste di connessione

OpenSSL s_server

Mediante il comando **s_client** è possibile connettersi al server creato al passo precedente, sulla porta 12345

```
openssl s_client –msg –connect localhost:12345
```

Alla ricezione di una richiesta di connessione effettuata mediante **s_client**, l'**s_server** mostrerà un output simile a quello seguente

```
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIAOQQgU/gS+ul8meX1MVvGTnfdrc/ddc0gXnqFsk+yE3c
sT/ME
MLYSu2pX8cv9ZJH29UfuxDkNzUYU7oopAvPybF/iEBzZfj68SzLkh7LL/Fw
FpXog
JKEGAgRYux70ogQCAgEspAYEBAEAAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers:DHE-RSA-AES256-SHA:DHE-DSS-AES256-
SHA:AES256-SHA:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-
SHA:DES-CBC3-SHA:DHE-RSA-AES128-SHA:DHE-DSS-AES128-
SHA:AES128-SHA:DHE-RSA-SEED-SHA:DHE-DSS-SEED-SHA:SEED-
SHA:RC4-SHA:RC4-MD5:EDH-RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-
SHA:DES-CBC-SHA:EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-DSS-DES-
CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-CBC-MD5:EXP-RC4-MD5
CIPHER is DHE-RSA-AES256-SHA
Secure Renegotiation IS supported
```

OpenSSL s_server

Server

```
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIAOQQgU/gS+ul8meX1MVvGTnfdr
c/ddc0gXnqFsk+yE3csT/ME
MLYSu2pX8cv9ZJH29UfuxDkNzUYU7oopAvPybF/iE
BzZfj68SzLkh7LL/FwFpXog
JKEGAgRYux70ogQCAgEspAYEBAEAAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers:DHE-RSA-AES256-SHA:DHE-
DSS-AES256-SHA:AES256-SHA:EDH-RSA-DES-
CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DES-CBC3-
SHA:DHE-RSA-AES128-SHA:DHE-DSS-AES128-
SHA:AES128-SHA:DHE-RSA-SEED-SHA:DHE-DSS-
SEED-SHA:SEED-SHA:RC4-SHA:RC4-MD5:EDH-
RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-SHA:DES-
CBC-SHA:EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-
DSS-DES-CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-
CBC-MD5:EXP-RC4-MD5
CIPHER is DHE-RSA-AES256-SHA
Secure Renegotiation IS supported
ciao
prova
```

Client

```
New, TLSv1/SSLv3, Cipher is DHE-RSA-
AES256-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : DHE-RSA-AES256-SHA
    Session-ID:
53F812FAE97C99E5F5315BC64E77DDADCFDD75CD2
05E7A85B24FB213772C4FF3
    Session-ID-ctx:
    Master-Key:
B612BB6A57F1CBFD6491F6F547EEC4390DCD4614E
E8A2902F3F26C5FE2101CD97E3EBC4B32E487B2CB
FC5C05A57A2024
    Key-Arg   : None
    Start Time: 1488658164
    Timeout   : 300 (sec)
    Verify return code: 21 (unable to
verify the first certificate)
---
ciao
prova
```

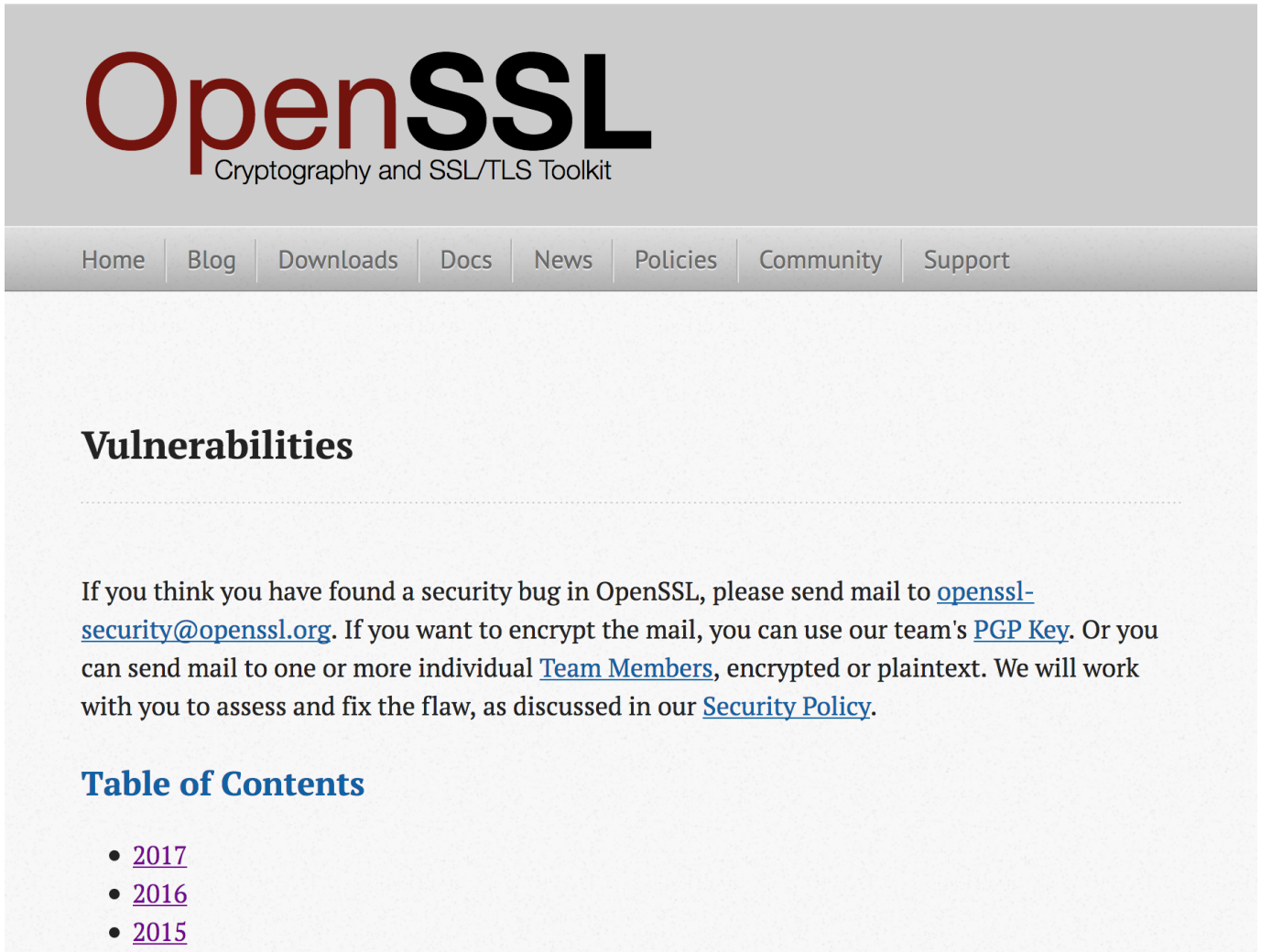
Dopo che la sessione sicura è stata stabilita, è possibile inviare messaggi, dal client al server e viceversa

OpenSSL s_server

Mediante il comando **s_server** è anche possibile creare un basilare web server, a cui è possibile connettersi mediante browser

```
openssl s_server -accept 12345 -cert utente1-cert.pem -key
private/utente1-key.pem -www
```

Vulnerabilità di OpenSSL



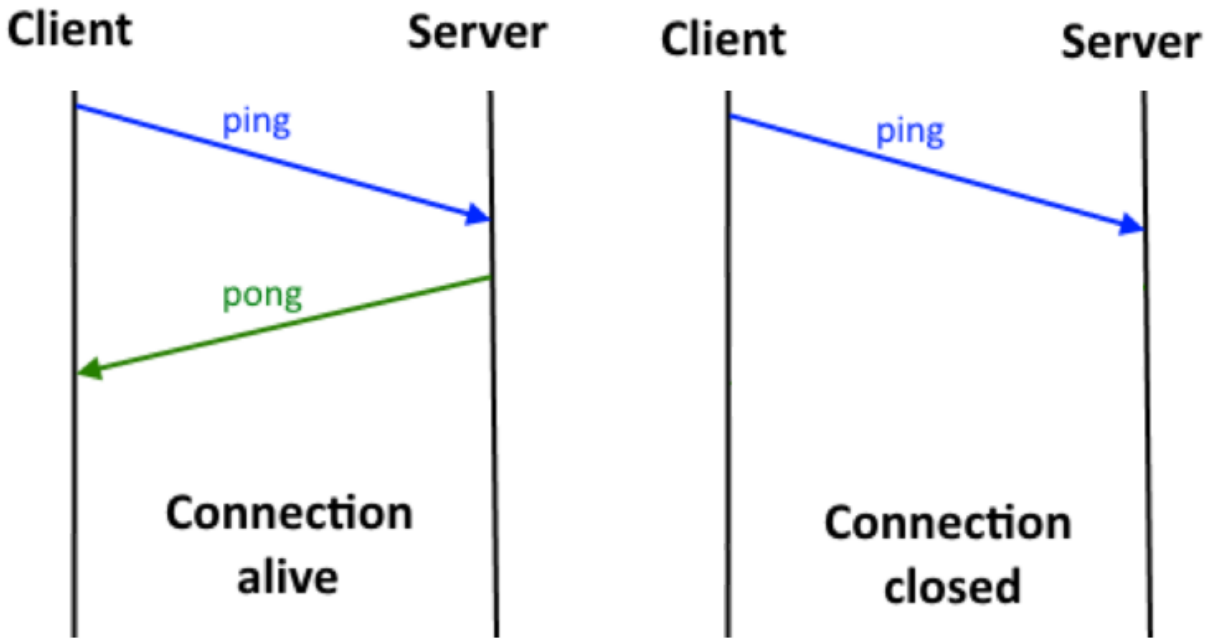
https://www.openssl.org/news/vulnerabilities.html

Importante vulnerabilità di OpenSSL (2014)

Il bug Heartbleed

Heartbleed
(Heartbeat extension)

- Meccanismo di *keep-alive* tra due host (client e server) implementato nella libreria OpenSSL



Heartbleed
(Timeline)

Feb 2012 – Release TLS/DTLS Heartbeat Extension.

07 Apr 2014 – Scoperta di Heartbleed Bug e Rilascio OpenSSL versione 1.0.1g

19 Apr 2014 – Prime dichiarazioni ufficiali

Heartbleed
(Bug)

- **Heartbleed** rappresenta un gioco di parole che sta per
  - **Heart** si riferisce al protocollo "Heartbeat"
  - **bleed** "sanguinare", in riferimento alla perdita di dati
- Segnalato da un gruppo di ricercatori finlandesi della società *Codenomicon* e quasi in contemporanea da due ingegneri di Google
- Sfruttando tale bug, si entra in possesso dei dati presenti in memoria RAM del server al momento dell'attacco
- Per maggiori dettagli
  - http://heartbleed.com/

Heartbleed
(Bollettino)

```
OpenSSL Security Advisory [07 Apr 2014]
=======================================

TLS heartbeat read overrun (CVE-2014-0160)
==========================================

A missing bounds check in the handling of the TLS heartbeat extension can be
used to reveal up to 64k of memory to a connected client or server.

Only 1.0.1 and 1.0.2-beta releases of OpenSSL are affected including
1.0.1f and 1.0.2-beta1.

Thanks for Neel Mehta of Google Security for discovering this bug and to
Adam Langley <agl@chromium.org> and Bodo Moeller <bmoeller@acm.org> for
preparing the fix.

Affected users should upgrade to OpenSSL 1.0.1g. Users unable to immediately
upgrade can alternatively recompile OpenSSL with -DOPENSSL_NO_HEARTBEATS.

1.0.2 will be fixed in 1.0.2-beta2.
```

https://www.openssl.org/news/secadv/20140407.txt

Heartbleed
(Intervista a Robin Seggelmann)

Sviluppatore tedesco che ha scritto la parte del codice affetto dal bug.

"*Stavo contribuendo al progetto OpenSSL*".

"*L'errore è conseguenza dell'introduzione di una nuova feature*." (TLS Heartbeat, da cui il nome del bug).

"*Non si tratta di un errore concettuale, ma implementativo*".

[Intervista al portale Wired, 10 Aprile 2014]

Heartbleed
(Società Colpite)

Dati aggiornati al 19 Aprile 2014

Nome	Colpito	Applicata Patch	Richiesta modifica password
Facebook	Non chiarito	Si	Si
Instagram	Si	Si	Si
LinkedIn	No	No	No
Twitter	No	Si	Non dichiarato
Apple	No	No	No
Amazon	No	No	No
Google	Si	Si	Si
Microsoft	No	No	No
Yahoo	Si	Si	Si
Youtube	Si	Si	Si

Heartbleed
(Hardware Affetto)

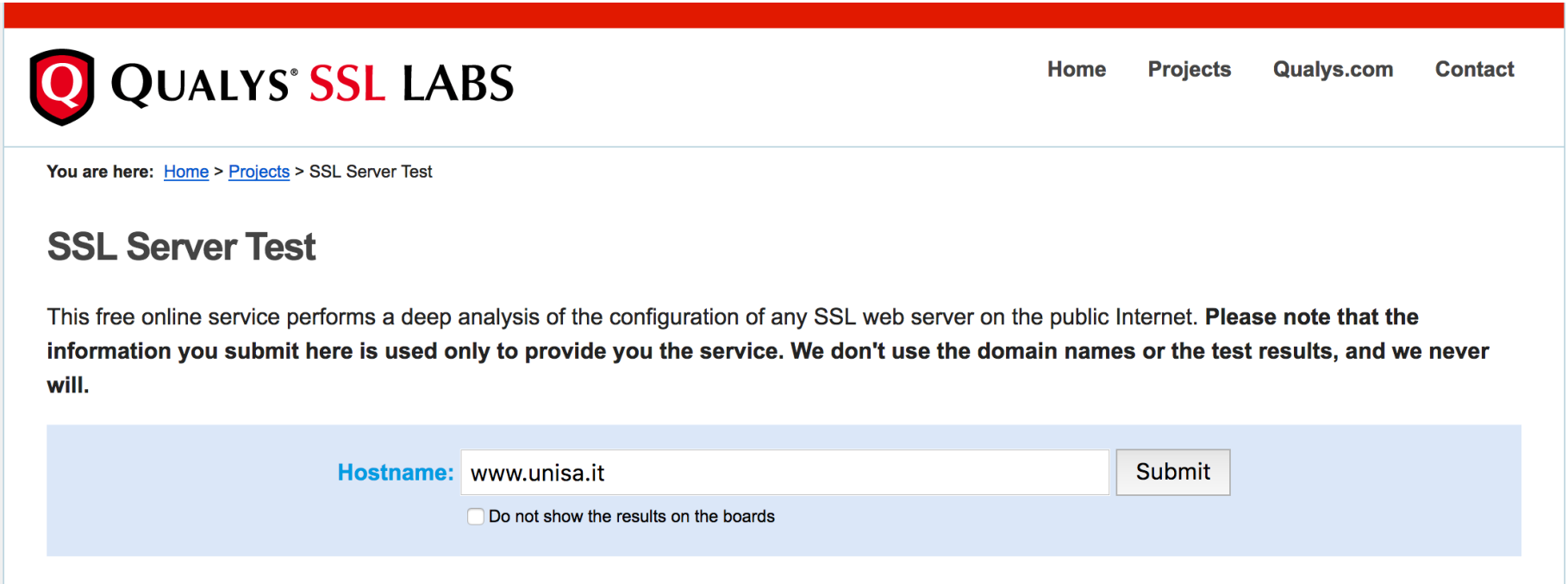
- Heartbleed Bug non riguarda solo il web, ma anche dispositivi quali
  - Smartphone con sistema operativo Android 4.1.1 "JellyBean"
  - Router Cisco
  - Router Juniper
  - Firmware della famiglia di prodotti "Western Digital My Cloud"

Heartbleed
(Sistemi Operativi Affetti)

- Esempi di sistemi operativi affetti (tra parentesi le relative date di rilascio)
  - Debian "Wheezy" (stable, Maggio 2013)
  - Ubuntu 12.04.4 LTS (Aprile 2012)
  - CentOS 6.5, OpenSSL (Gennaio 2013)
  - OpenBSD 5.3 (Maggio 2012)
  - FreeBSD 10.0 (Gennaio 2014)
  - NetBSD 5.0.2 (Aprile 2009)
  - OpenSuse 12.2 (Maggio 2012)

SSL Server Test

https://www.ssllabs.com/ssltest/index.html



SSL Server Test

https://www.ssllabs.com/ssltest/index.html

Certificate #1: RSA 2048 bits (SHA256withRSA)

Server Key and Certificate #1

Subject	www.unisa.it Fingerprint SHA1: 2c47710c67dfee5239cfc0c852575374648e5373 Pin SHA256: Z5Wyl+sAr2OvZDYqydi4qShab8zQHA41tjQD8FxdCrQ=
Common names	www.unisa.it
Alternative names	www.unisa.it www.personaldesk.unisa.it
Valid from	Wed, 12 Nov 2014 00:00:00 UTC
Valid until	Sat, 11 Nov 2017 23:59:59 UTC (expires in 6 months and 21 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	TERENA SSL CA 2 AIA: http://crt.usertrust.com/TERENASSLCA2.crt
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	No
OCSP Must Staple	No
Revocation information	CRL, OCSP CRL: http://crl.usertrust.com/TERENASSLCA2.crl OCSP: http://ocsp.usertrust.com
Revocation status	Good (not revoked)
DNS CAA	No (more info)
Trusted	Yes

SSL Server Test

https://www.ssllabs.com/ssltest/index.html

Additional Certificates (if supplied)

Certificates provided	4 (5220 bytes)
Chain issues	**Contains anchor**

#2

Subject	TERENA SSL CA 2 Fingerprint SHA1: 38525c7140d285040e02dd2a7f3c7dba21042e01 Pin SHA256: PYHJ7Ok9y2OoV3yMZFAcH45HI64yll/qcT9kRYmQFTY=
Valid until	Tue, 08 Oct 2024 23:59:59 UTC (expires in 7 years and 5 months)
Key	RSA 2048 bits (e 65537)
Issuer	USERTrust RSA Certification Authority
Signature algorithm	SHA384withRSA

#3

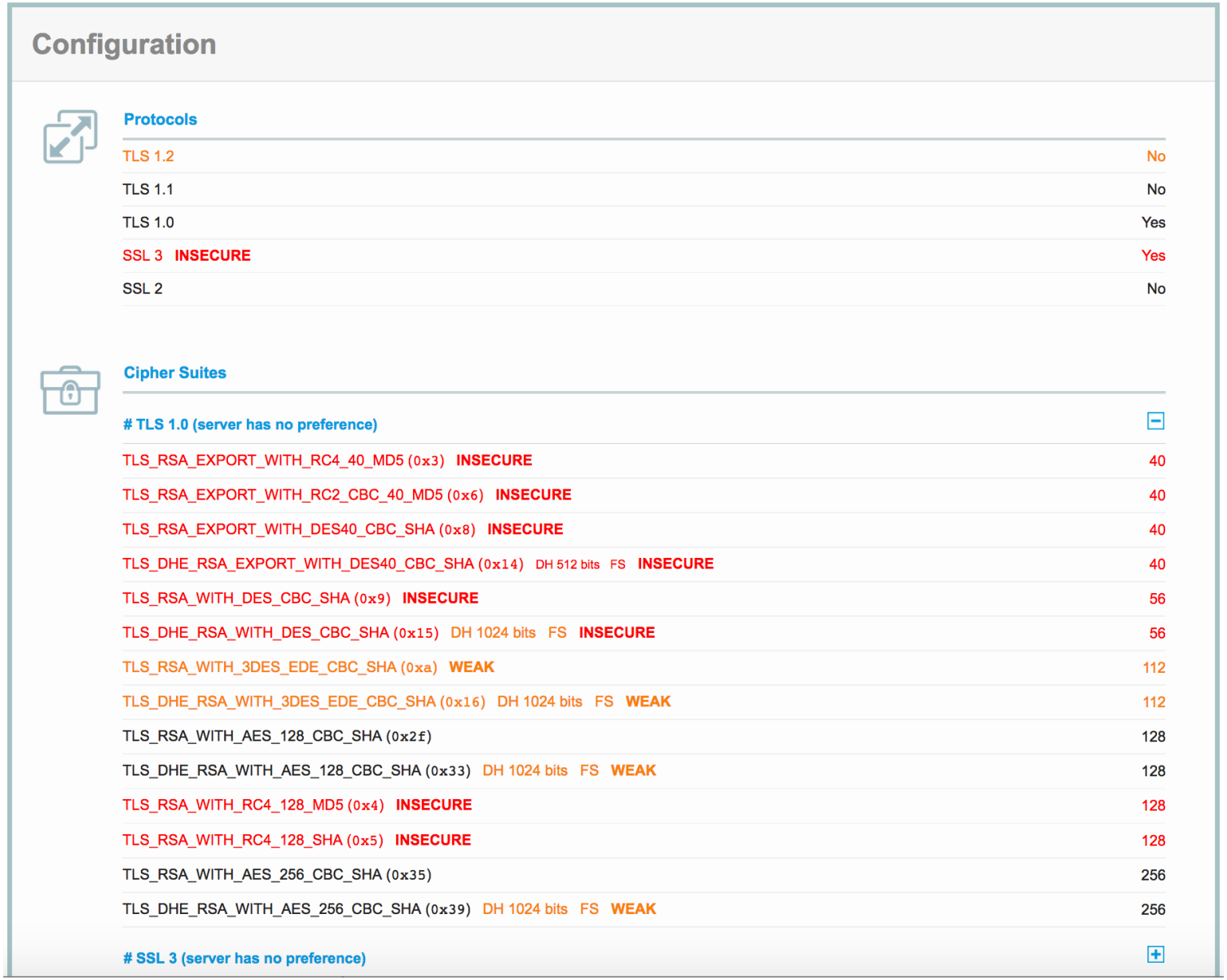
Subject	USERTrust RSA Certification Authority Fingerprint SHA1: eab040689a0d805b5d6fd654fc168cff00b78be3 Pin SHA256: x4QzPSC810K5/cMjb05Qm4k3Bw5zBn4ITdO/nEW/Td4=
Valid until	Sat, 30 May 2020 10:48:38 UTC (expires in 3 years and 1 month)
Key	RSA 4096 bits (e 65537)
Issuer	AddTrust External CA Root
Signature algorithm	SHA384withRSA

#4

Subject	AddTrust External CA Root In trust store Fingerprint SHA1: 02faf3e291435468607857694df5e45b68851868 Pin SHA256: ICppFqbkrlJ3EcVFAkeip0+44VaoJUymbnOaEUk7tEU=
Valid until	Sat, 30 May 2020 10:48:38 UTC (expires in 3 years and 1 month)
Key	RSA 2048 bits (e 65537)
Issuer	AddTrust External CA Root Self-signed
Signature algorithm	SHA1withRSA Weak, but no impact on root certificate

SSL Server Test

https://www.ssllabs.com/ssltest/index.html



Configuration

Protocols

TLS 1.2	No
TLS 1.1	No
TLS 1.0	Yes
SSL 3 **INSECURE**	Yes
SSL 2	No

Cipher Suites

# TLS 1.0 (server has no preference)	
TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x3) **INSECURE**	40
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x6) **INSECURE**	40
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA (0x8) **INSECURE**	40
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA (0x14) DH 512 bits FS **INSECURE**	40
TLS_RSA_WITH_DES_CBC_SHA (0x9) **INSECURE**	56
TLS_DHE_RSA_WITH_DES_CBC_SHA (0x15) DH 1024 bits FS **INSECURE**	56
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa) **WEAK**	112
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16) DH 1024 bits FS **WEAK**	112
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) DH 1024 bits FS **WEAK**	128
TLS_RSA_WITH_RC4_128_MD5 (0x4) **INSECURE**	128
TLS_RSA_WITH_RC4_128_SHA (0x5) **INSECURE**	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) DH 1024 bits FS **WEAK**	256
# SSL 3 (server has no preference)	

SSL Server Test

https://www.ssllabs.com/ssltest/index.html

Protocol Details

DROWN	No, server keys and hostname not seen elsewhere with SSLv2 **(1) For a better understanding of this test, please read this longer explanation** (2) Key usage data kindly provided by the Censys network search engine; original DROWN test here (3) Censys data is only indicative of possible key and certificate reuse; possibly out-of-date and not complete
Secure Renegotiation	**Supported**
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Not mitigated server-side (more info) SSL 3: 0x6, TLS 1.0: 0x6
POODLE (SSLv3)	**Vulnerable INSECURE** (more info) SSL 3: 0x6
POODLE (TLS)	No (more info)
Downgrade attack prevention	**Yes, TLS_FALLBACK_SCSV supported** (more info)
SSL/TLS compression	No
RC4	**Yes INSECURE** (more info)
Heartbeat (extension)	No
Heartbleed (vulnerability)	No (more info)
Ticketbleed (vulnerability)	No (more info)
OpenSSL CCS vuln. (CVE-2014-0224)	No (more info)
OpenSSL Padding Oracle vuln. (CVE-2016-2107)	No (more info)
Forward Secrecy	**Insecure key exchange INSECURE**
ALPN	No
NPN	No
Session resumption (caching)	Yes
Session resumption (tickets)	No
OCSP stapling	No
Strict Transport Security (HSTS)	No

SSL Server Test

https://www.ssllabs.com/ssltest/index.html

HSTS Preloading	Not in: Chrome Edge Firefox IE
Public Key Pinning (HPKP)	No (more info)
Public Key Pinning Report-Only	No
Public Key Pinning (Static)	No (more info)
Long handshake intolerance	No
TLS extension intolerance	No
TLS version intolerance	No
Incorrect SNI alerts	No
Uses common DH primes	**Yes** Replace with custom DH parameters if possible (more info)
DH public server param (Ys) reuse	No
ECDH public server param reuse	No, ECDHE suites not supported
Supported EC Named Curves	-
SSL 2 handshake compatibility	Yes

HTTP Requests

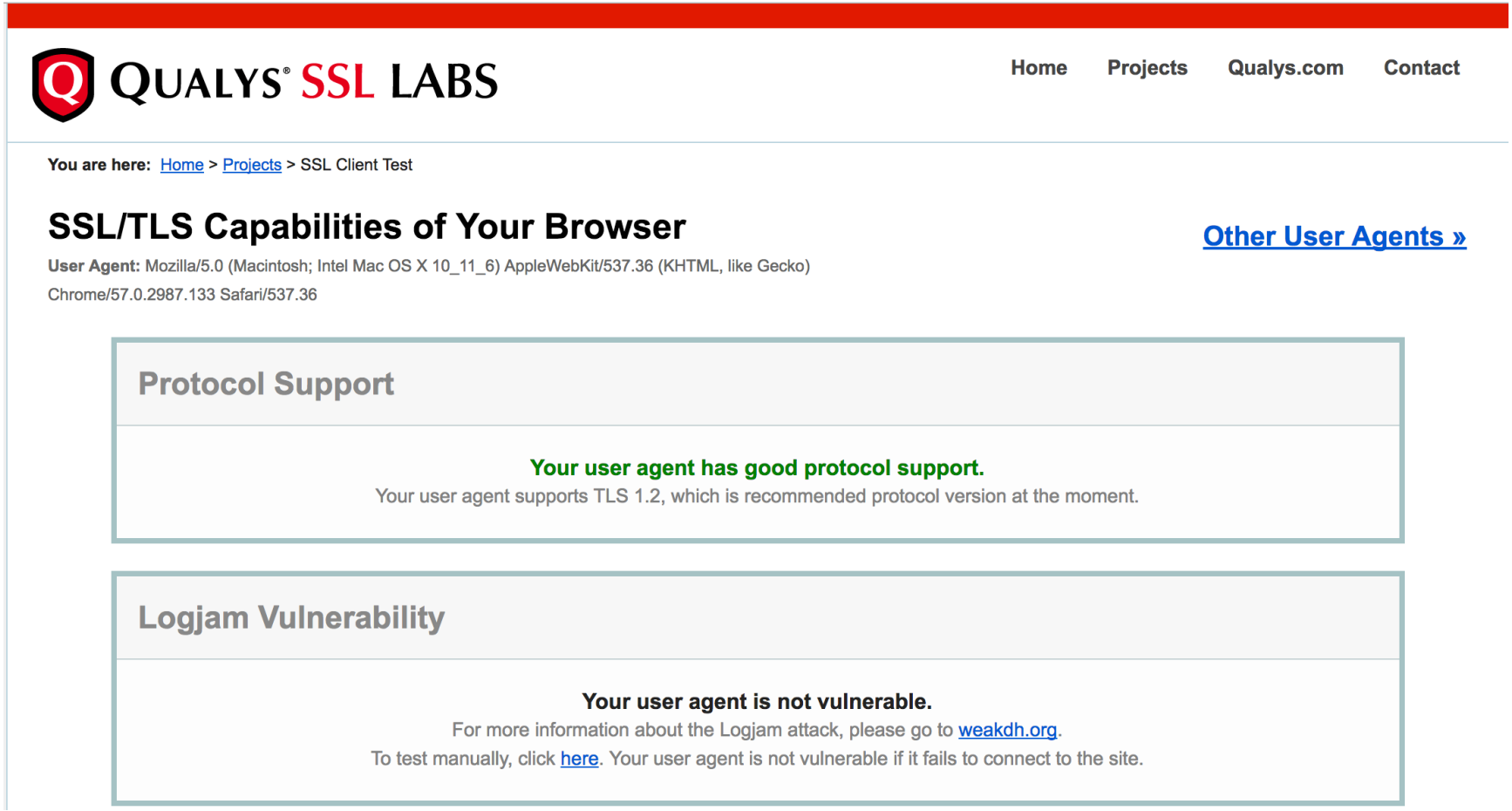
1 **https://www.unisa.it/** (HTTP/1.1 200 OK)

Miscellaneous

Test date	Fri, 21 Apr 2017 16:28:05 UTC
Test duration	95.41 seconds
HTTP status code	200
HTTP server signature	Apache/2.2.3 (CentOS)
Server hostname	www3.unisa.it

SSL Client Test

https://www.ssllabs.com/ssltest/viewMyClient.html



SSL Client Test

https://www.ssllabs.com/ssltest/viewMyClient.html

FREAK Vulnerability

Your user agent is not vulnerable.

For more information about the FREAK attack, please go to www.freakattack.com.

To test manually, click here. Your user agent is not vulnerable if it fails to connect to the site.

POODLE Vulnerability

Your user agent is not vulnerable.

For more information about the POODLE attack, please read this blog post.

Protocol Features

Protocols	
TLS 1.3	No
TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

SSL Client Test

https://www.ssllabs.com/ssltest/viewMyClient.html

Cipher Suites (in order of preference)

TLS_GREASE_1A (0x1a1a)	-
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) Forward Secrecy	128
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) Forward Secrecy	256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) Forward Secrecy	256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) Forward Secrecy	256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) Forward Secrecy	256
OLD_TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc14) Forward Secrecy	256
OLD_TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcc13) Forward Secrecy	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) Forward Secrecy	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) Forward Secrecy	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	128
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	256
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa) **WEAK**	168

(1) When a browser supports SSL 2, its SSL 2-only suites are shown only on the very first connection to this site. To see the suites, close all browser windows, then open this exact page directly. Don't refresh.

SSL Client Test

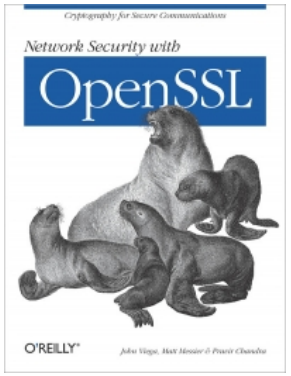
https://www.ssllabs.com/ssltest/viewMyClient.html

Protocol Details

Server Name Indication (SNI)	Yes
Secure Renegotiation	Yes
TLS compression	No
Session tickets	Yes
OCSP stapling	Yes
Signature algorithms	SHA256/ECDSA, RSA_PSS_SHA256, SHA256/RSA, SHA384/ECDSA, RSA_PSS_SHA384, SHA384/RSA, RSA_PSS_SHA512, SHA512/RSA, SHA1/RSA
Elliptic curves	tls_grease_4a4a, x25519, secp256r1, secp384r1
Next Protocol Negotiation	No
Application Layer Protocol Negotiation	Yes h2 http/1.1
SSL 2 handshake compatibility	No

Bibliografia

- Network Security with OpenSSL
  Pravir Chandra, Matt Messier and John Viega (2002), O'Reilly
  - Cap. 1
  - Appendix A. Command-Line Reference



- Documentazione su OpenSSL
  - https://www.openssl.org/docs/

OpenSSL
Cryptography and SSL/TLS Toolkit

Domande?