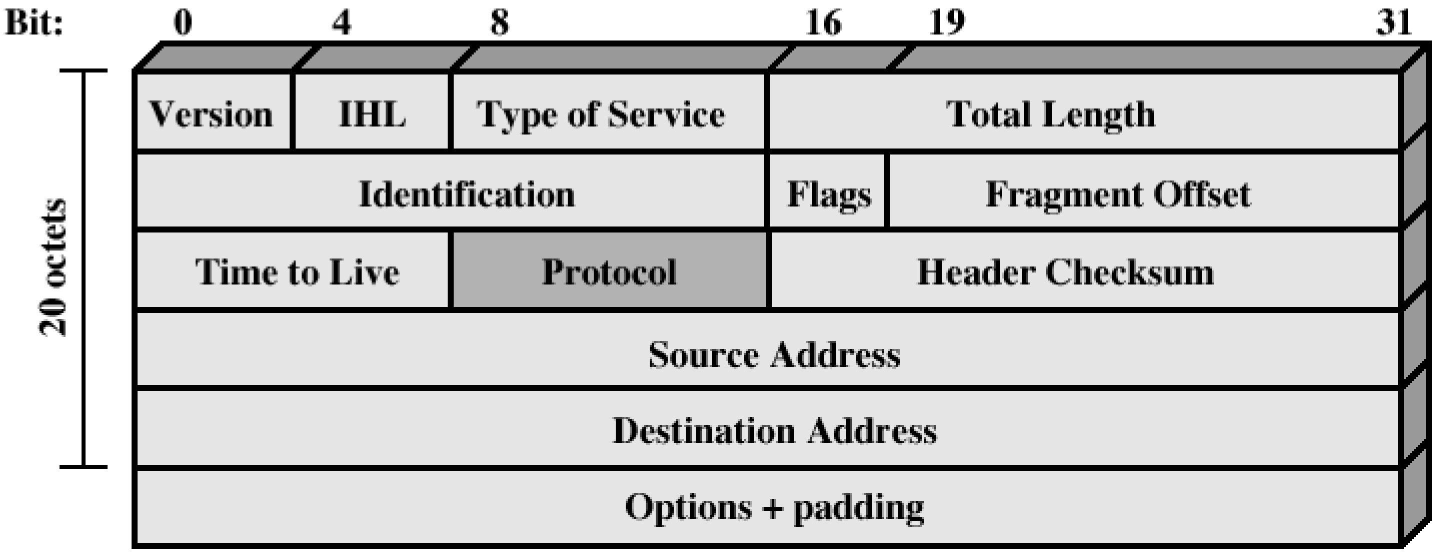
6669 Criptografía y Seguridad Informática

Seguridad en IP: TLS

Ing Hugo Pagola

IPv4



Bit: 0 4 8 16 19 31

20 octets

Version | IHL | Type of Service | Total Length

Identification | Flags | Fragment Offset

Time to Live | Protocol | Header Checksum

Source Address

Destination Address

Options + padding

(a) IPv4 Header

Los datos [payload] siguen al encabezado

¿Es seguro IP?

Los datos (Payload) no viajan cifrados

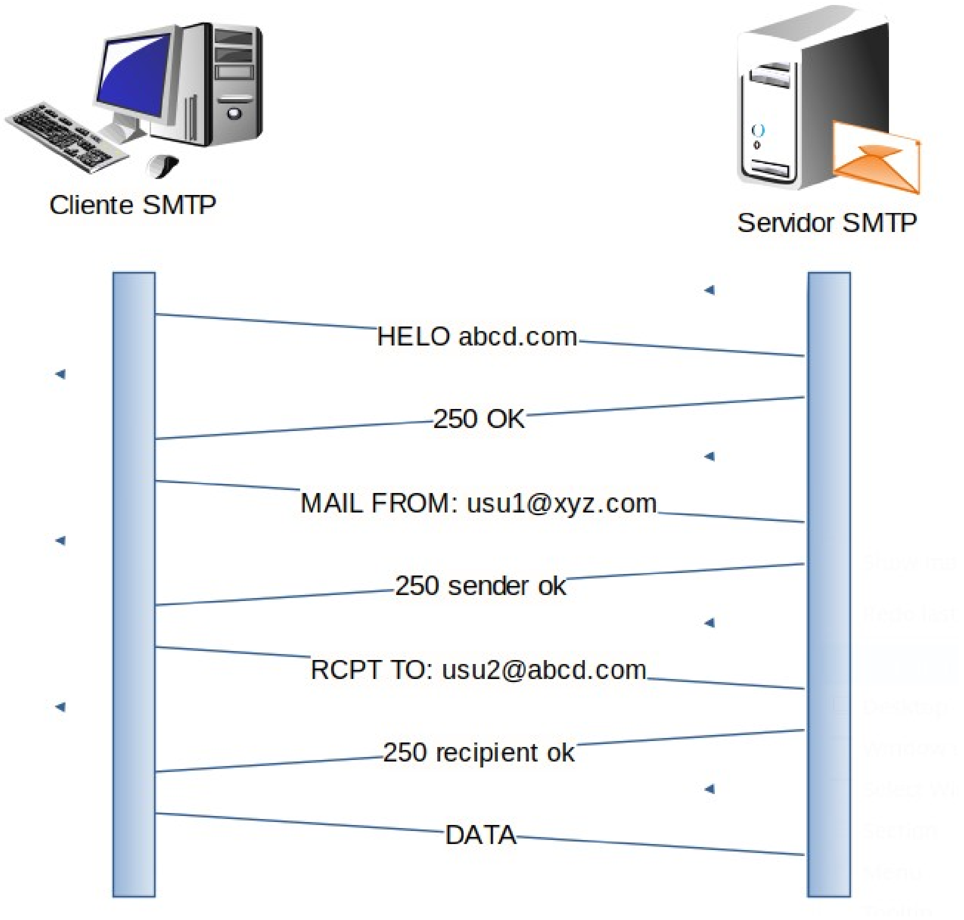
- No se ofrece confidencialidad
- IP sniffers están disponibles en la red
- Las direcciones IP pueden ser observadas
- La autenticación basada en direcciones IP addresses puede ser rota

=> IP tiene algunas debilidades

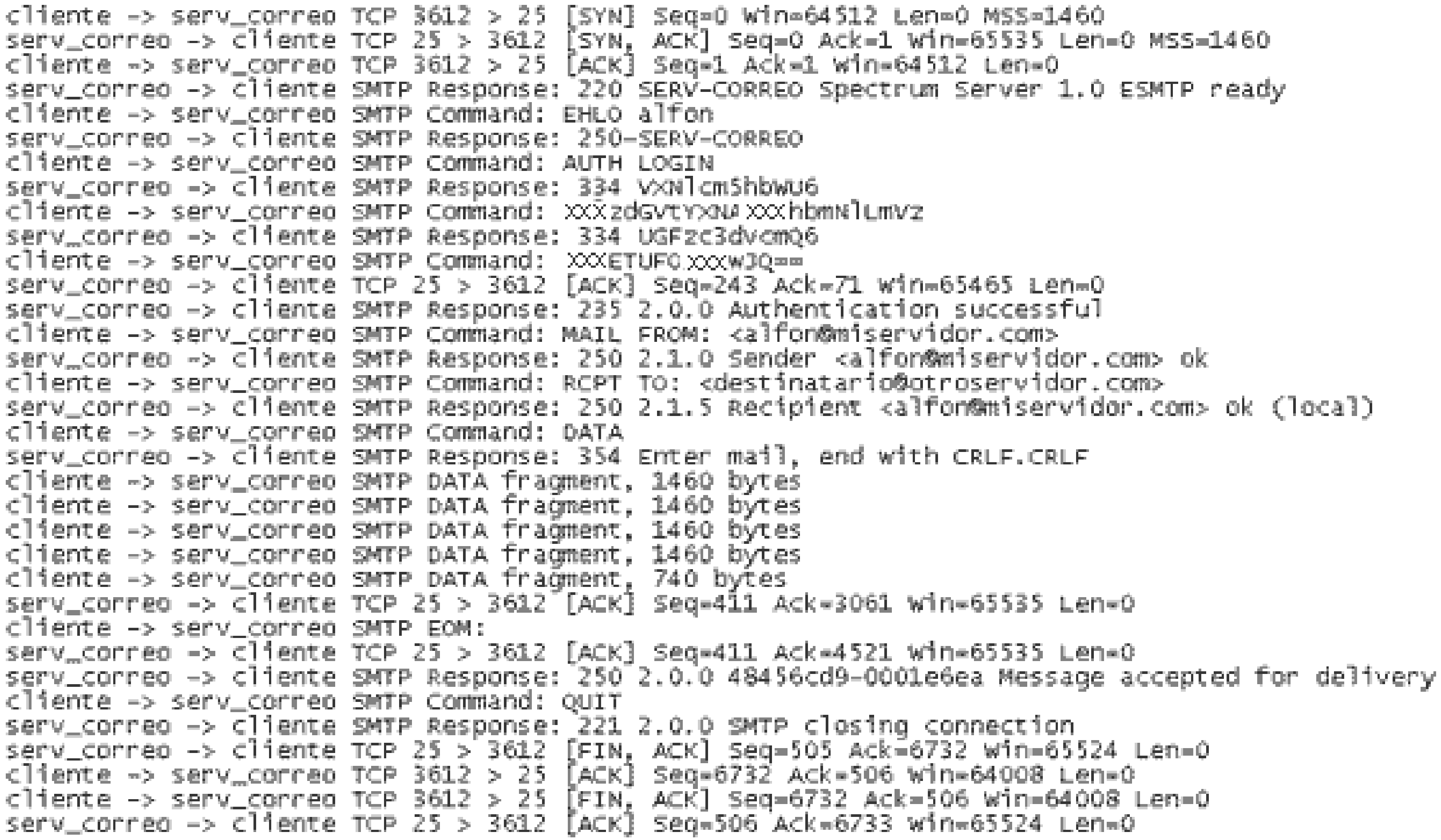
SMTP: Simple Mail Transfer Protocol

RFC 821

El cliente se conecta al puerto 25



Tshark SMTP

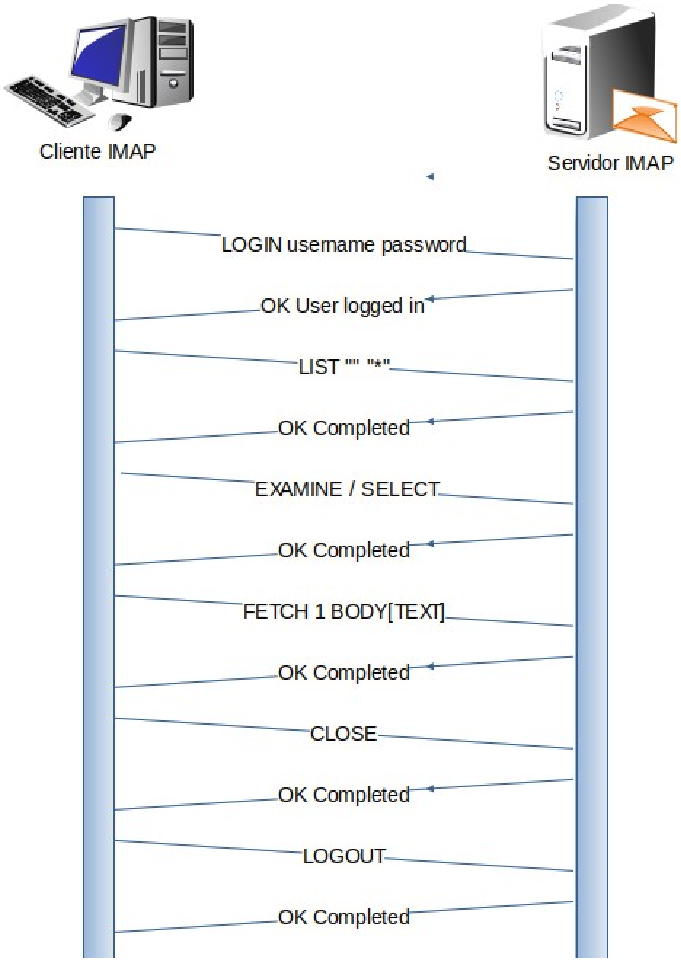


```
cliente -> serv_correo TCP 3612 > 25 [SYN] Seq=0 Win=64512 Len=0 MSS=1460
serv_correo -> cliente TCP 25 > 3612 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
cliente -> serv_correo TCP 3612 > 25 [ACK] Seq=1 Ack=1 Win=64512 Len=0
serv_correo -> cliente SMTP Response: 220 SERV-CORREO Spectrum Server 1.0 ESMTP ready
cliente -> serv_correo SMTP Command: EHLO alfon
serv_correo -> cliente SMTP Response: 250-SERV-CORREO
cliente -> serv_correo SMTP Command: AUTH LOGIN
serv_correo -> cliente SMTP Response: 334 VXNlcm5hbWU6
cliente -> serv_correo SMTP Command: XXXzdGVtYXNA XXXhbmNlLmVz
serv_correo -> cliente SMTP Response: 334 UGFzc3dvcmQ6
cliente -> serv_correo SMTP Command:  XXXETUFO XXXwJQ==
serv_correo -> cliente TCP 25 > 3612 [ACK] Seq=243 Ack=71 Win=65465 Len=0
serv_correo -> cliente SMTP Response: 235 2.0.0 Authentication successful
cliente -> serv_correo SMTP Command: MAIL FROM: <alfon@miservidor.com>
serv_correo -> cliente SMTP Response: 250 2.1.0 Sender <alfon@miservidor.com> ok
cliente -> serv_correo SMTP Command: RCPT TO: <destinatario@otroservidor.com>
serv_correo -> cliente SMTP Response: 250 2.1.5 Recipient <alfon@miservidor.com> ok (local)
cliente -> serv_correo SMTP Command: DATA
serv_correo -> cliente SMTP Response: 354 Enter mail, end with CRLF.CRLF
cliente -> serv_correo SMTP DATA fragment, 1460 bytes
cliente -> serv_correo SMTP DATA fragment, 1460 bytes
cliente -> serv_correo SMTP DATA fragment, 1460 bytes
cliente -> serv_correo SMTP DATA fragment, 1460 bytes
cliente -> serv_correo SMTP DATA fragment, 740 bytes
serv_correo -> cliente TCP 25 > 3612 [ACK] Seq=411 Ack=3061 Win=65535 Len=0
cliente -> serv_correo SMTP EOM:
serv_correo -> cliente TCP 25 > 3612 [ACK] Seq=411 Ack=4521 Win=65535 Len=0
serv_correo -> cliente SMTP Response: 250 2.0.0 48456cd9-0001e6ea Message accepted for delivery
cliente -> serv_correo SMTP Command: QUIT
serv_correo -> cliente SMTP Response: 221 2.0.0 SMTP closing connection
serv_correo -> cliente TCP 25 > 3612 [FIN, ACK] Seq=505 Ack=6732 Win=65524 Len=0
cliente -> serv_correo TCP 3612 > 25 [ACK] Seq=6732 Ack=506 Win=64008 Len=0
cliente -> serv_correo TCP 3612 > 25 [FIN, ACK] Seq=6732 Ack=506 Win=64008 Len=0
serv_correo -> cliente TCP 25 > 3612 [ACK] Seq=506 Ack=6733 Win=65524 Len=0
```

IMAP: Internet Message Access Protocol

RFC 3501

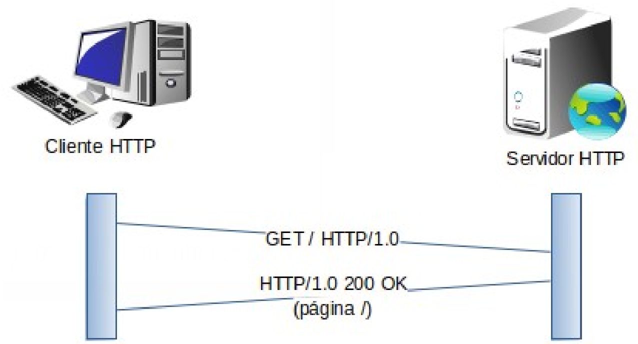
El cliente se conecta al puerto 143

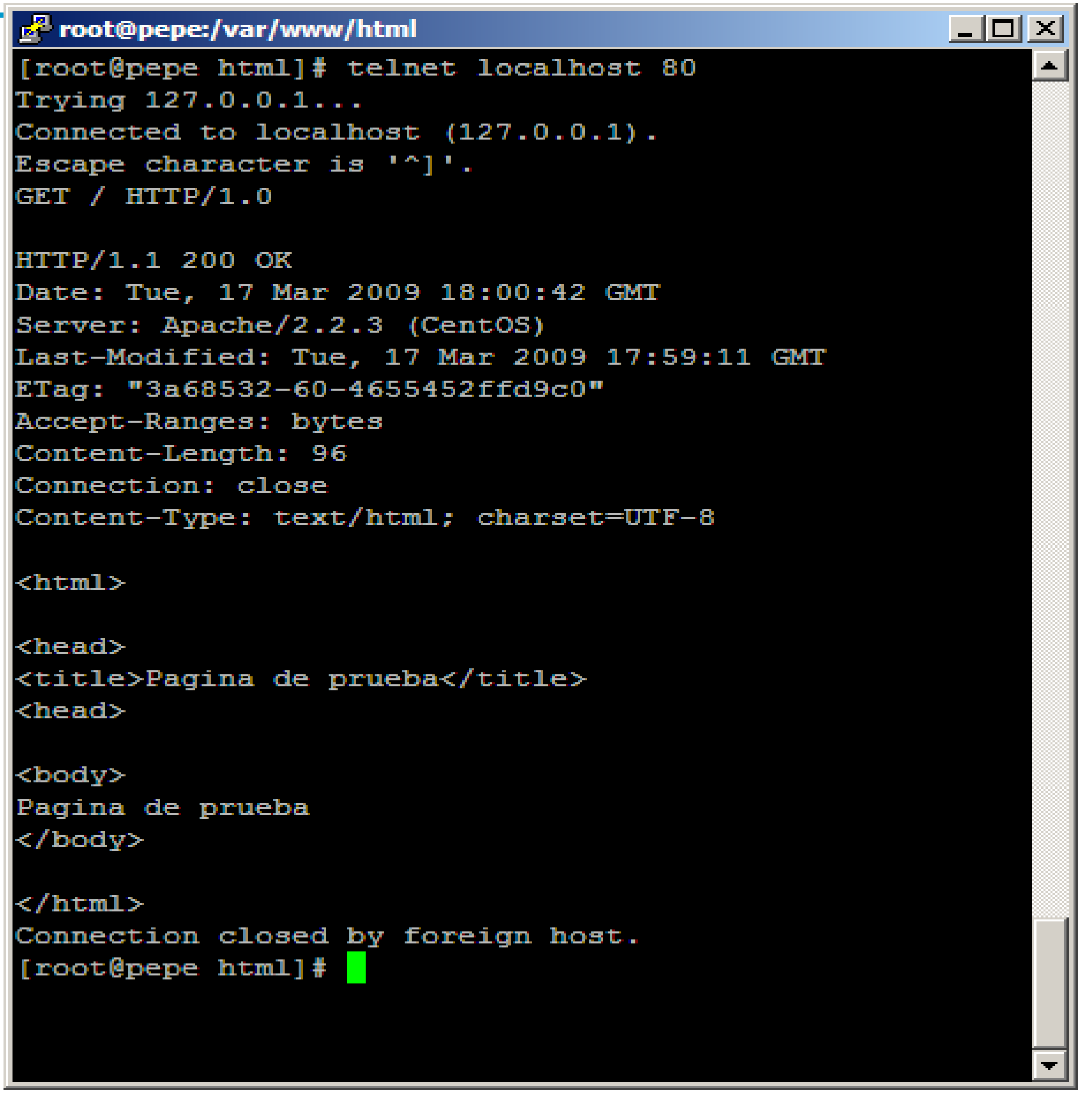


IMAP: Internet Message Access Protocol

```
# telnet 10.2.1.51 143
Trying 10.2.1.51...
Connected to 10.2.1.51 (10.2.1.51).
Escape character is '^]'.
* OK mail.abc.com.ar IMAP4rev1 service ready
a01 LOGIN miUsuario miPassword
a01 OK [CAPABILITY IMAP4rev1 ACL BINARY CATENATE CHILDREN CONDSTORE ENABLE ESEARCH ID
IDLE LIST-EXTENDED LITERAL+ LOGIN-REFERRALS MULTIAPPEND NAMESPACE QRESYNC QUOTA
RIGHTS=ektx SASL-IR SEARCHRES UIDPLUS UNSELECT WITHIN] LOGIN completed
a02 LIST "" "*"
* LIST (\HasChildren) "/" "INBOX"
* …
* LIST (\HasNoChildren) "/" "Trash"
a02 OK LIST completed
a03 SELECT INBOX
* 2114 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1] UIDs are valid for this mailbox
* OK [UIDNEXT 5690] next expected UID is 5690
* FLAGS (\Answered \Deleted \Draft \Flagged \Seen $Forwarded $MDNSent Forwarded $Junk
$NotJunk Junk JunkRecorded NonJunk NotJunk Importantes)
* OK [PERMANENTFLAGS (\Answered \Deleted \Draft \Flagged \Seen $Forwarded $MDNSent
Forwarded Importantes \*)] junk-related flags are not permanent
* OK [HIGHESTMODSEQ 10382] modseq tracked on this mailbox
a03 OK [READ-WRITE] SELECT completed
a04 FETCH 2114 BODY[TEXT]
* 2114 FETCH (BODY[TEXT] {146}
prueba

----------------------------------------------------------------
This message was sent using IMP, the Internet Messaging Program.


)
a04 OK FETCH completed
a05 CLOSE
a05 OK CLOSE completed
a06 LOGOUT
* BYE mail.abc.com.ar IMAP4rev1 server terminating connection
a06 OK LOGOUT completed
Connection closed by foreign host.
```

HTTP: Hypertext Transfer Protocol





```
[root@pepe html]# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 17 Mar 2009 18:00:42 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 17 Mar 2009 17:59:11 GMT
ETag: "3a68532-60-4655452ffd9c0"
Accept-Ranges: bytes
Content-Length: 96
Connection: close
Content-Type: text/html; charset=UTF-8

<html>

<head>
<title>Pagina de prueba</title>
<head>

<body>
Pagina de prueba
</body>

</html>
Connection closed by foreign host.
[root@pepe html]#
```

¿Dónde proveer Seguridad?

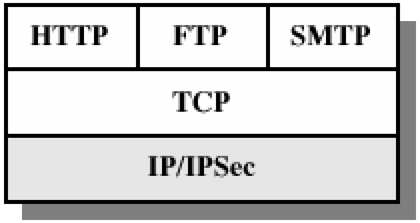
¿Capa de aplicación?

- S/MIME, PGP – seguridad en correo electrónico
- Kerberos – modelo cliente-servidor
- SSH –telnet seguro

¿Nivel de transporte?

- SSL / TLS
- Entre TCP y Aplicación

Nivel IP IPSec

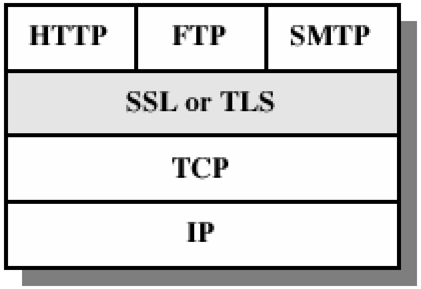


HTTP | FTP | SMTP

TCP

IP/IPSec

(a) Network Level



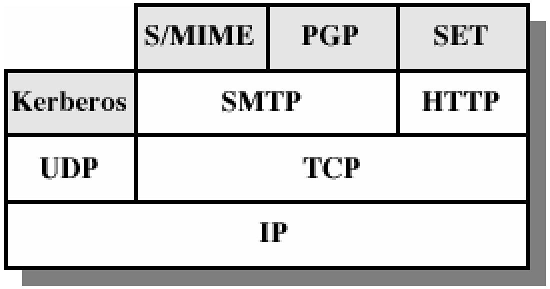
HTTP | FTP | SMTP

SSL or TLS

TCP

IP

(b) Transport Level



S/MIME | PGP | SET

Kerberos | SMTP | HTTP

UDP | TCP

IP

(c) Application Level

Primitivas Criptográficas

Cifradores

- Simétricos: AES
- Clave Publica: RSA

Hash: sha-2 (SHA256, sha384)

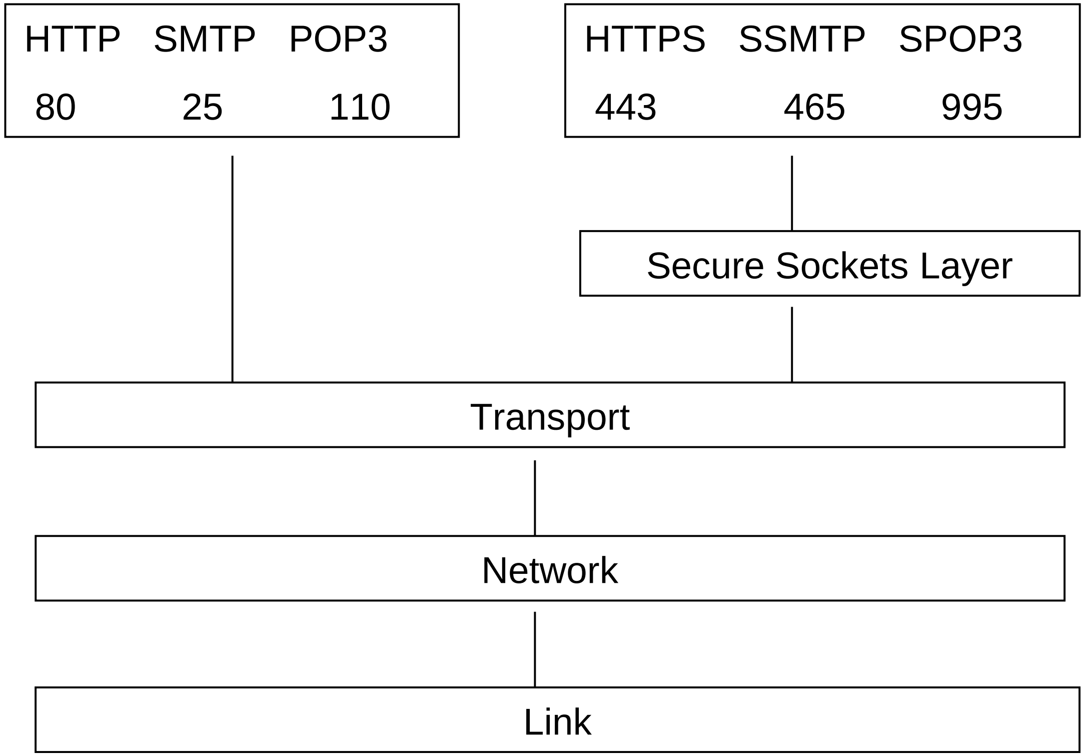
Autenticación Mensajes HMAC

Autenticación: challenge and response

Key agreement: Diffie-Hellman, ECDHE

Certificados

Firma Digital: RSA



HTTP SMTP POP3

80 25 110

HTTPS SSMTP SPOP3

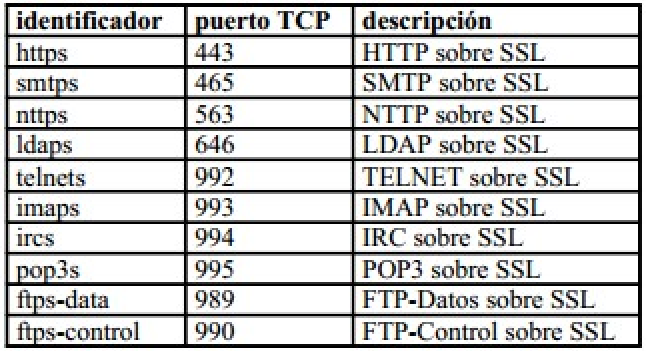
443 465 995

Secure Sockets Layer

Transport

Network

Link



identificador	puerto TCP	descripción
https	443	HTTP sobre SSL
smtps	465	SMTP sobre SSL
nttps	563	NTTP sobre SSL
ldaps	646	LDAP sobre SSL
telnets	992	TELNET sobre SSL
imaps	993	IMAP sobre SSL
ircs	994	IRC sobre SSL
pop3s	995	POP3 sobre SSL
ftps-data	989	FTP-Datos sobre SSL
ftps-control	990	FTP-Control sobre SSL

Seguridad en IP: TLS

Protocolo Transport Layer Security

Transport Layer Security (TLS)

- Versión actualizada de **SSL** (Secure Sockets Layer)
  - última versión de SSL (Netscape) fue 3.0 (1995)
  - TLS se identifica como SSL v 3.1
  - Similar, pero no compatible directamente.
- Protege una sesión entre **cliente** y **servidor**.
  - Típicamente, HTTP (navegador y web server).
- Requiere protocolo de transporte confiable(TCP)
- El grupo de trabajo de TLS (Transport Layer Security) [**https://tlswg.org/**

Historia

- TLSv1.0, IETF (RFC 2246, 1999)
    - Estándar IETF basado en SSLv3
    - Tenia otros 2 protocolos similares pero incompatibles el mas conocido fue shttp://
- TLSv1.1, IETF (RFC 4346, 2006)
- TLSv1.2, IETF (RFC 5246, 2008)
- TLSv1.3: RFC 8446 (2018)

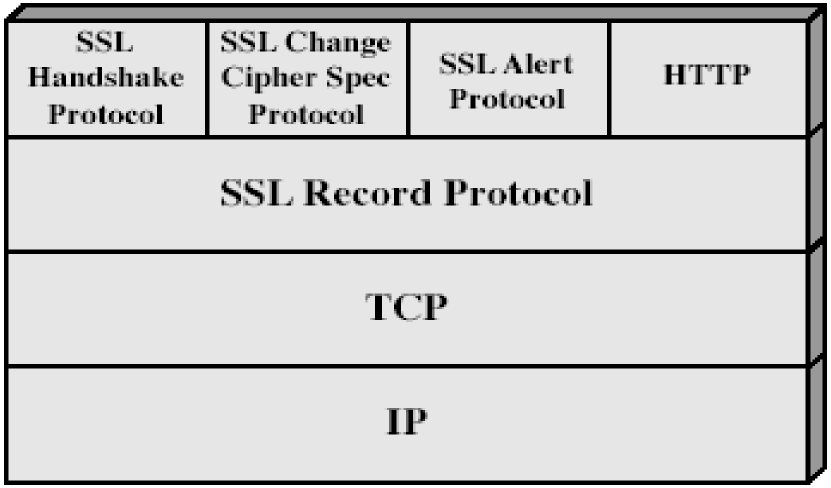
Servicios de Seguridad

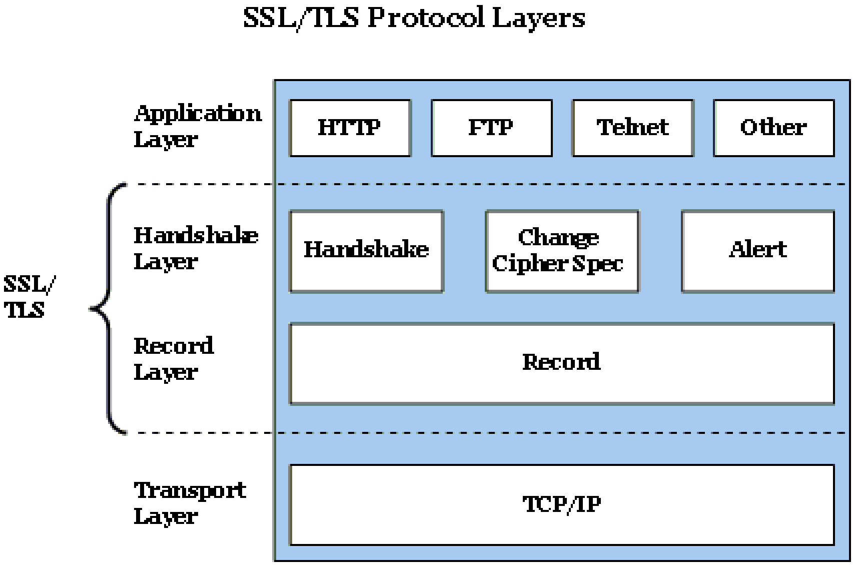
- Autenticación
  - del servidor frente al cliente;
  - opcionalmente, del cliente frente al servidor.
  - Mediante **certificados** de clave pública.

- Integridad
  - Mediante **MAC** y números de secuencia.

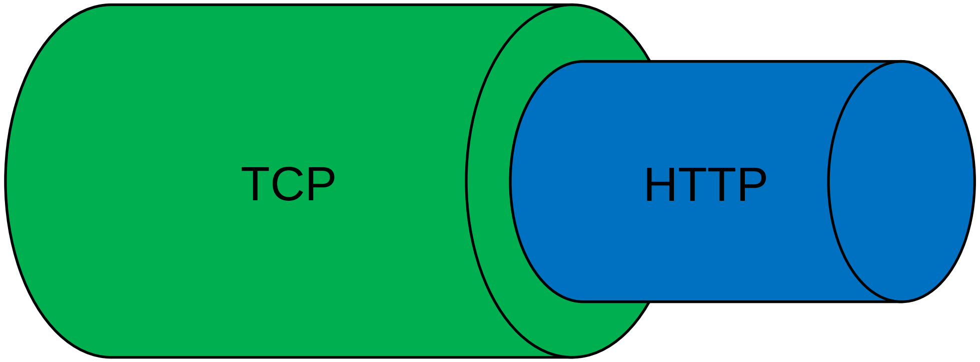
- Confidencialidad
  - Opcional: Mediante **cifrado** con algoritmo simétrico

Arquitectura de protocolos SSL

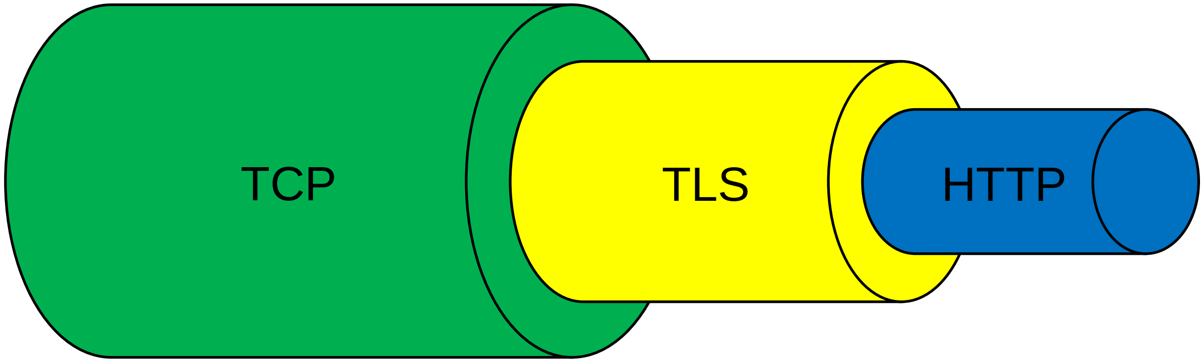




HTTP



HTTPS



SSL Record Protocol

fragmenta (en bloques de 2^{14} bytes) y opcionalmente comprime datos

Confidencialidad

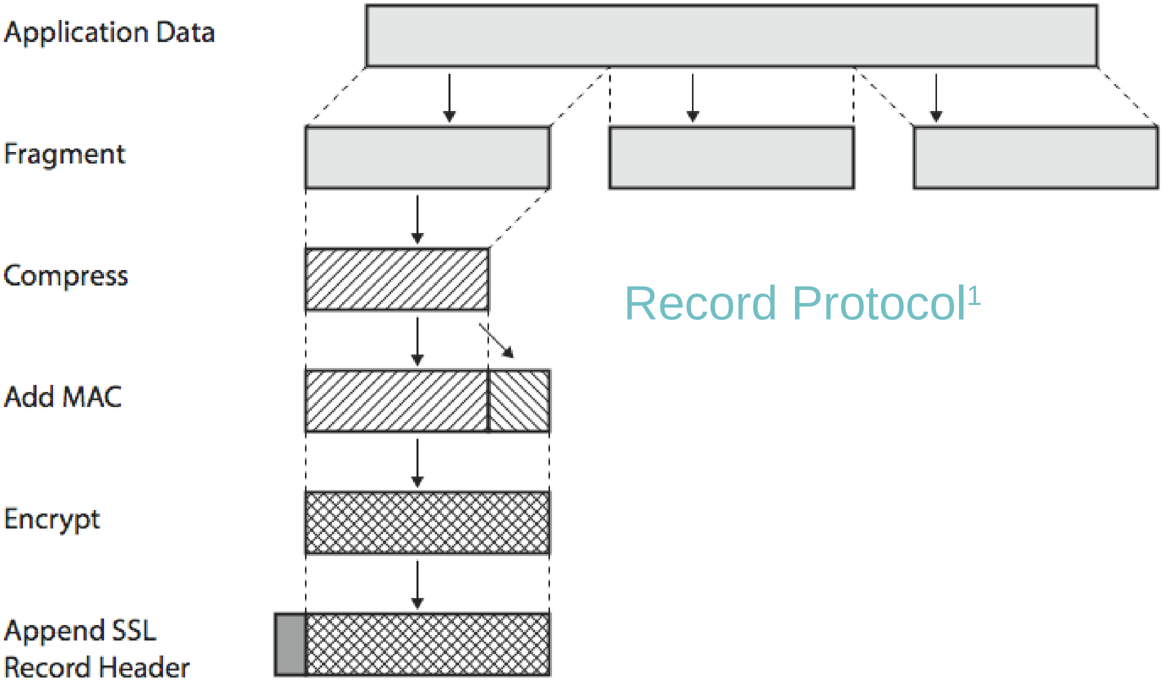
- Usa cifrado simétrico con una clave compartida definida por el protocolo Handshake.
- AES
- Opcionalmente, los mensajes son comprimidos.

Integridad

- Usa un MAC con claves compartidas
- Similar al HMAC pero con diferente padding (la clave es concatenada al mensaje en vez de XORed)

Record Protocol

- Fragmentación
- Compresión (optional)
- Calculo del MAC (Message Authentication Code)
- Se lo cifra

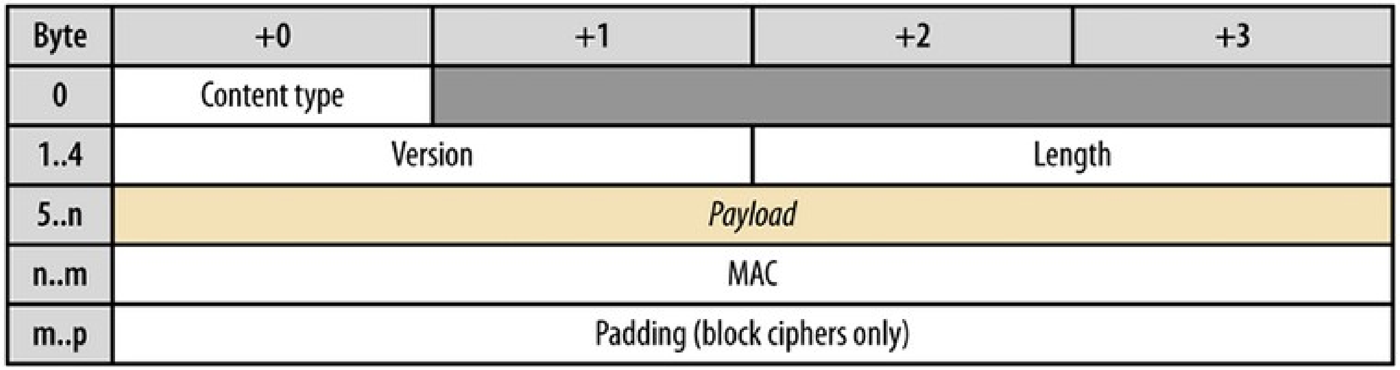


Record Protocol[1]

Record Protocol – Cont.

- Record Protocol Header
  - Content Type
    - ChangeCipherSpec
    - Alert
    - Handshake
    - Application
  - Major/minor version
  - Length of plaintext (or compressed fragment)
  - Message itself
  - MAC
  - Padding

Record Protocol Packet[6]



Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

Protocolo “SSL Alert”

Notifica de las alertas relacionadas con el SSL.

Severidad

- warning or fatal

Alertas especificas

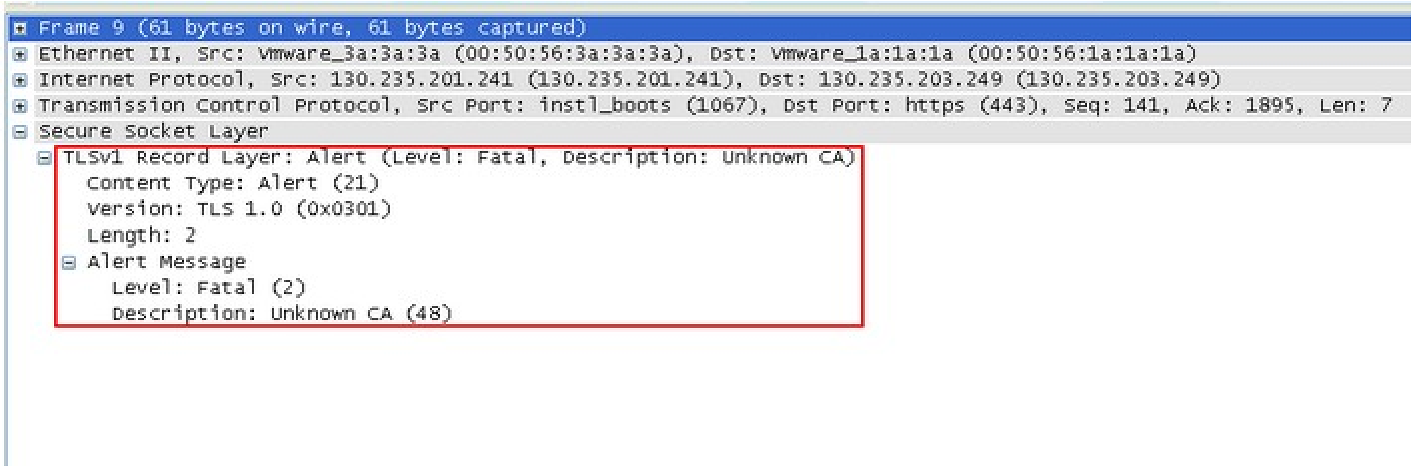
- unexpected message
- bad record mac
- decompression failure, handshake failure, illegal parameter
- close notify,
- no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown

Compresión y encriptado como el resto del trafico

Protocolo “SSL Alert”

Cada mensaje tiene 2 bytes

- Un byte para el nivel de seguridad (severidad)
    - warning (La conexión puede reanudarse)
    - fatal (La conexión se termina inmediatamente)
- Un byte para el código de alerta
    - Falla en el intercambio parámetros ilegales. Sin certificado, mal certificado, certificado no soportado, certificado revocado, certificado expirado, certificado desconocido



```
⊞ Frame 9 (61 bytes on wire, 61 bytes captured)
⊞ Ethernet II, Src: Vmware_3a:3a:3a (00:50:56:3a:3a:3a), Dst: Vmware_1a:1a:1a (00:50:56:1a:1a:1a)
⊞ Internet Protocol, Src: 130.235.201.241 (130.235.201.241), Dst: 130.235.203.249 (130.235.203.249)
⊞ Transmission Control Protocol, Src Port: instl_boots (1067), Dst Port: https (443), Seq: 141, Ack: 1895, Len: 7
⊟ Secure Socket Layer
  ⊟ TLSv1 Record Layer: Alert (Level: Fatal, Description: Unknown CA)
      Content Type: Alert (21)
      Version: TLS 1.0 (0x0301)
      Length: 2
    ⊟ Alert Message
        Level: Fatal (2)
        Description: Unknown CA (48)
```

Seguridad en IP: TLS

Protocolo “Handshake”

Fases de TLS

Una comunicación a través de TLS implica tres fases:

- **Establecimiento de la conexión y negociación de los algoritmos criptográficos** que van a usarse en la comunicación, a partir del conjunto de algoritmos soportados por cada uno de los interlocutores.

- **Intercambio de claves**, empleando algún mecanismo de clave pública y autentificación de los interlocutores a partir de sus certificados digitales.

- **Cifrado simétrico del tráfico**

Protocolo “SSL Handshake”

Este protocolo es ejecutado antes que cualquier intercambio de datos se realice

- Es decir, el protocolo record no comienza hasta que no acabe el de intercambio de datos.

De hecho, el protocolo de intercambio de datos (de manera abreviada) aun si una sesión anterior es relanzada

Arquitectura SSL

SSL session

- Asociación entre cliente y servidor
- Creado por el protocolo de Handshake.
- Define un conjunto de parámetros criptográficos.
- Puede ser compartido entre distintas conexiones SSL

SSL connection

- Transitorio, representa una conexión
- Esta asociado con una sesión SSL

Parámetros de estado de una Sesión

- Identificador de Sesión
  - Escogido por el servidor
- Certificado de contraparte
  - (certificado del servidor si la entidad es el cliente, o del cliente si la entidad es el servidor)
  - Puede ser *null* (probable para el server, si no se autentico a la contraparte)
- Método de Compresión
  - Algoritmo usado por compresión
- Especificación de cifrado
  - Algoritmos de cifrado por bloques
  - Algoritmo hash usado para integridad (SHA-1)
- Clave Maestra
  - 48-bytes secretos compartidos

- Es re-ejecutable
  - bandera que indica si la sesión puede ser reutilizada

Session State	
Session ID	**Arbitrary**
Peer Certificate	**Can be null**
Compression	**Typically null**
CipherSpec	**Cipher algorithm + hash algorithm + more**
Master Secret	**Secret key between client/server**
Resumable?	**Can use to make new connections**

Architecture – Cont.

- Conexion
  - Varias conexiones por sesion

Connection State	
Client and Server Random	Arbitrary bytes
Server write MAC secret	Secret key for MAC
Client write MAC secret	Secret key for MAC
Server write key	Symmetric key for decrypting data sent by server
Client write key	Symmetric key for decrypting data sent by client
IV’s	Use with CBC mode (Final Ciphertext block becomes IV for next record
Sequence Number	Sequence # for sent/received messages

Tipos de mensajes en el protocolo handshake

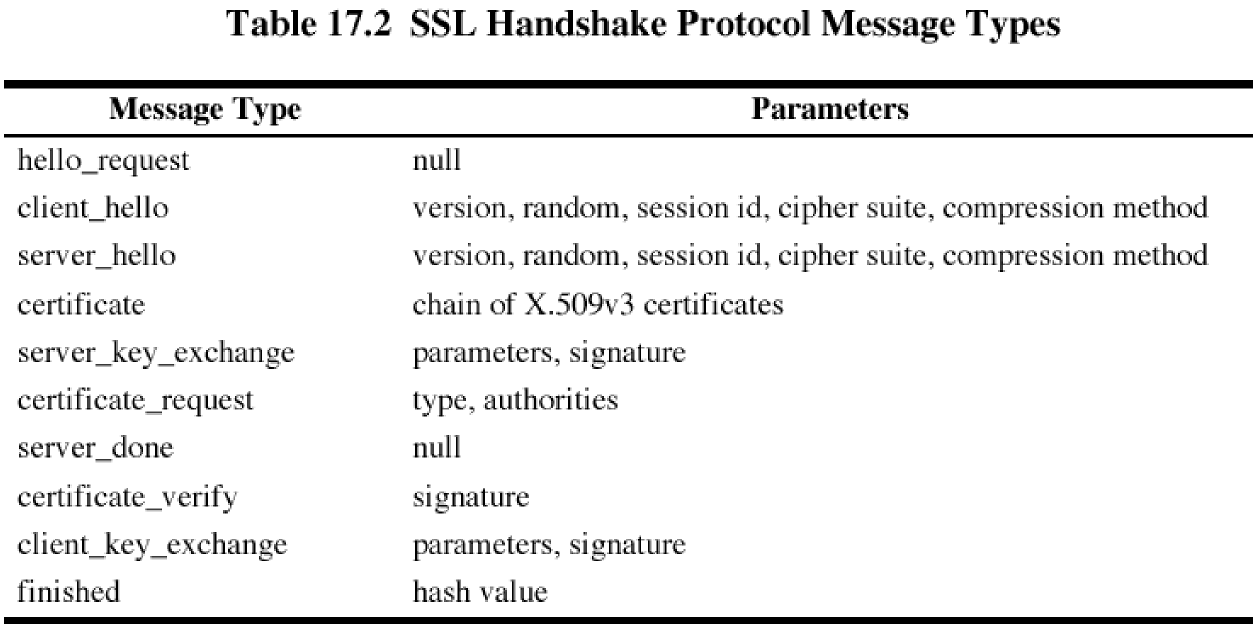
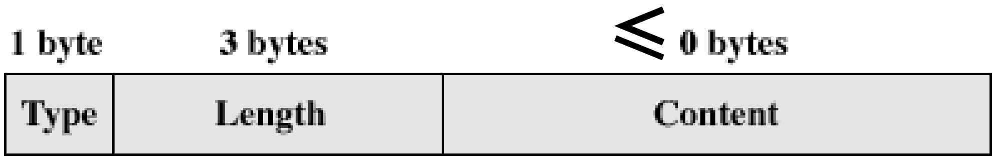


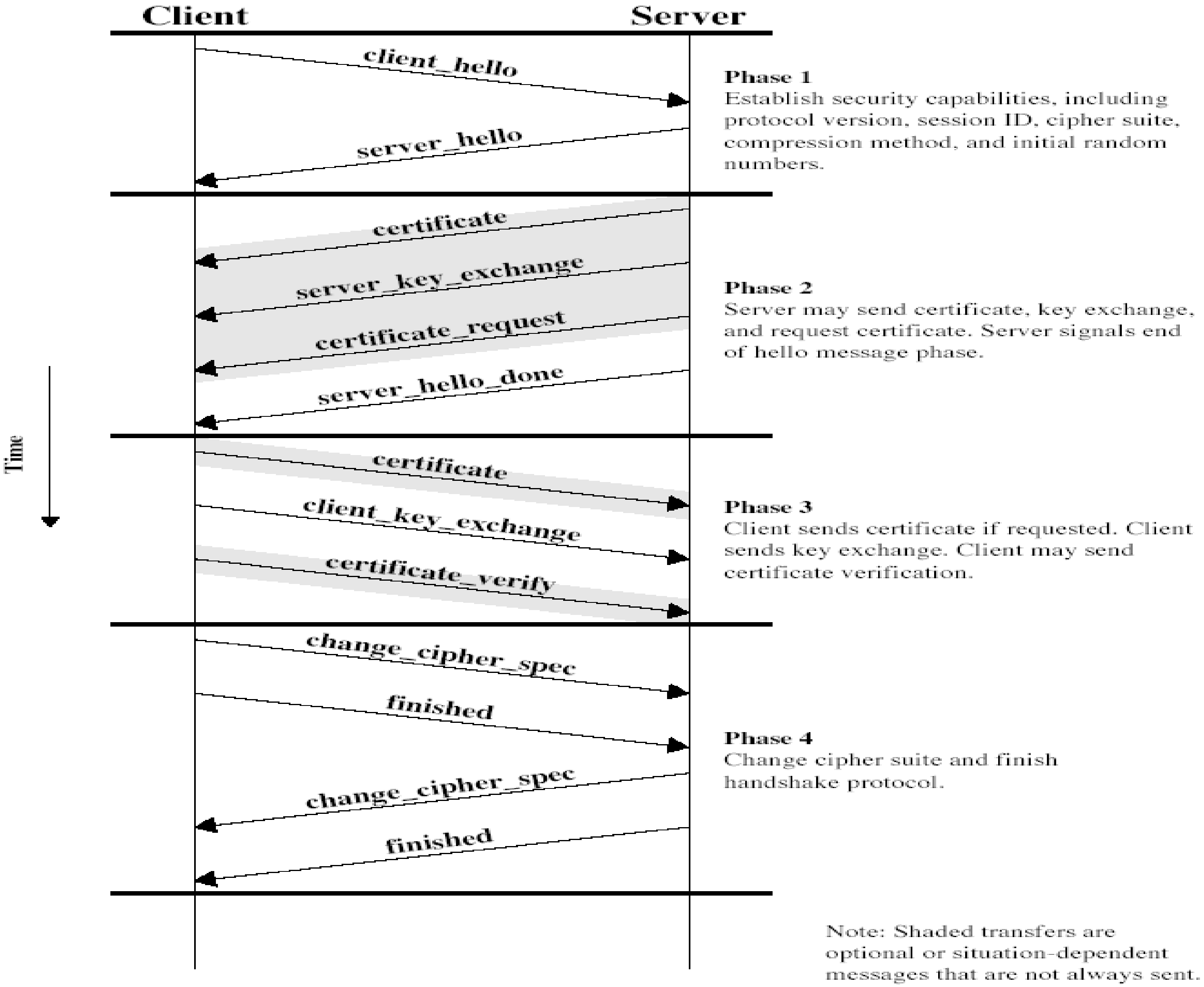
Table 17.2 SSL Handshake Protocol Message Types

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value



1 byte	3 bytes	⩽ 0 bytes
Type	Length	Content

Handshake Protocol



Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

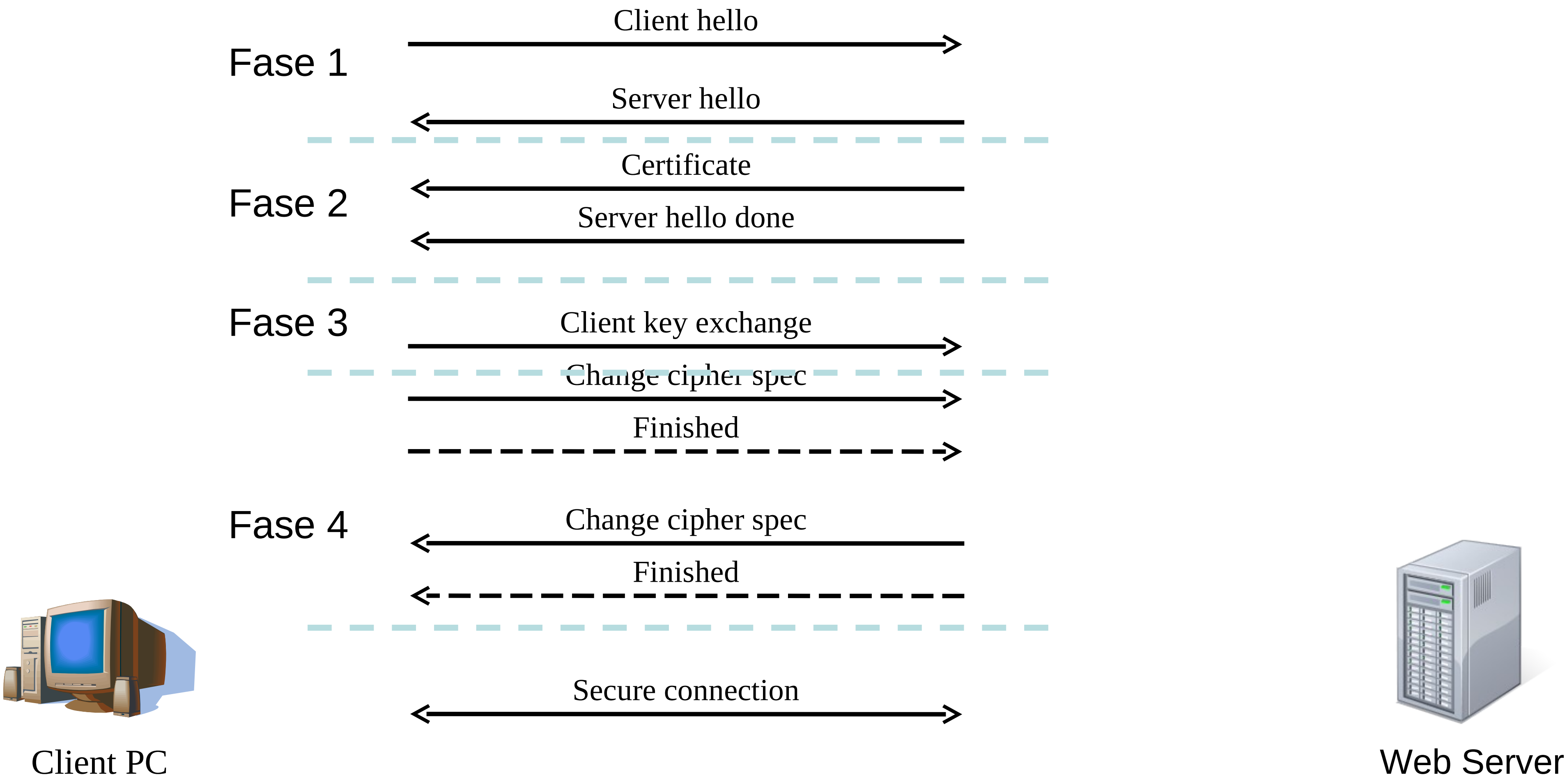
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

One Way



Fase 1

Client hello

Server hello

Fase 2

Certificate

Server hello done

Fase 3

Client key exchange

Change cipher spec

Finished

Fase 4

Change cipher spec

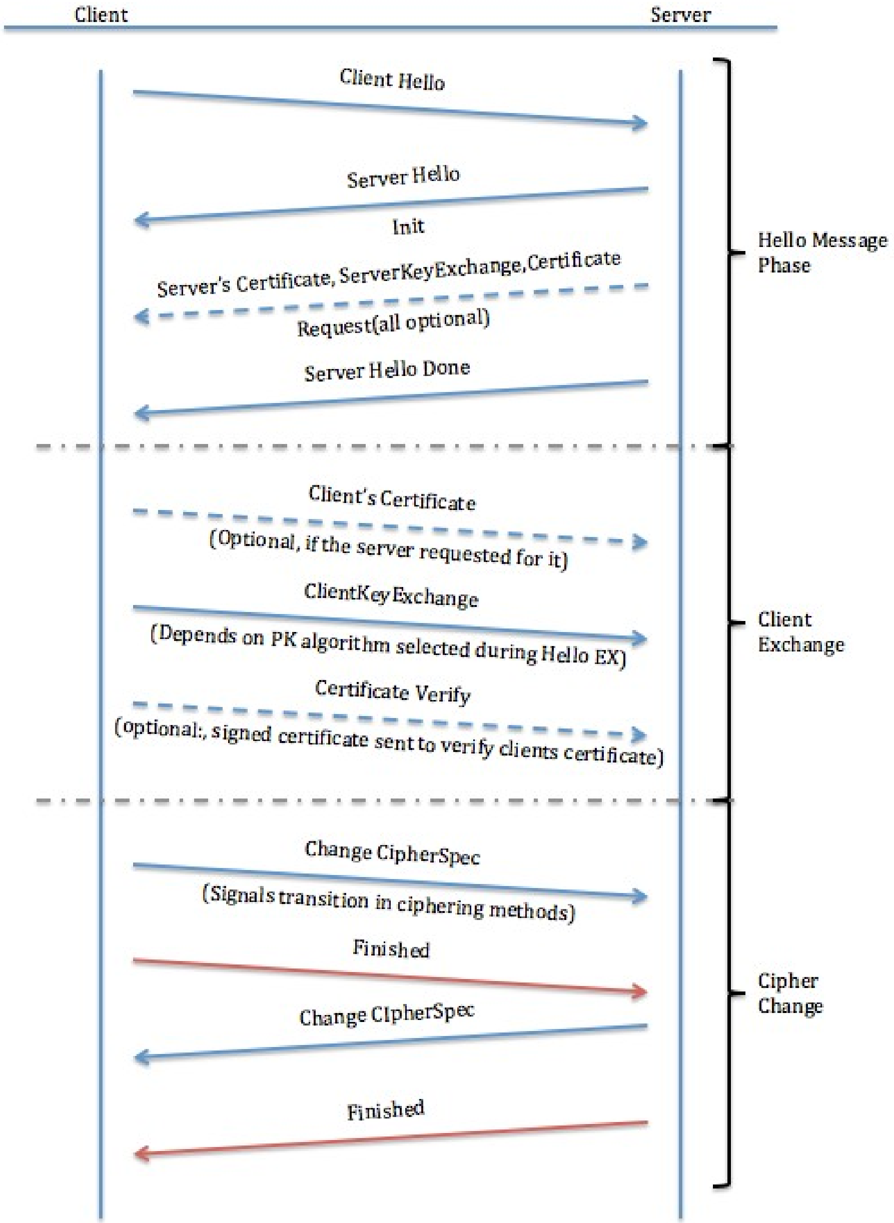
Finished

Secure connection

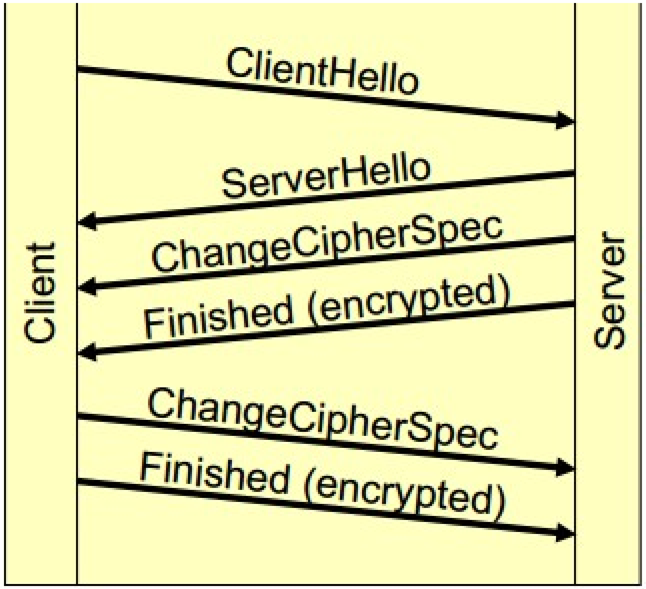
Client PC

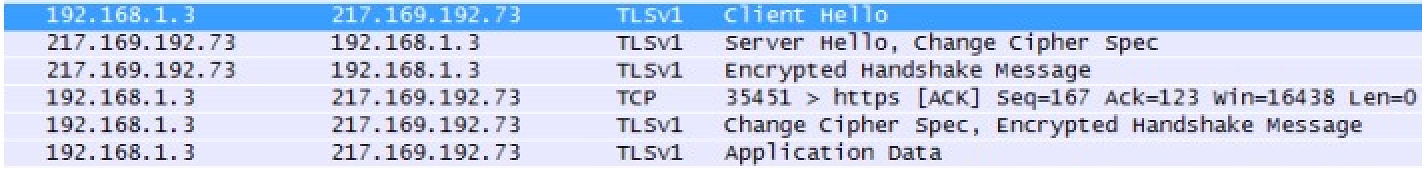
Web Server

Two way Handshake

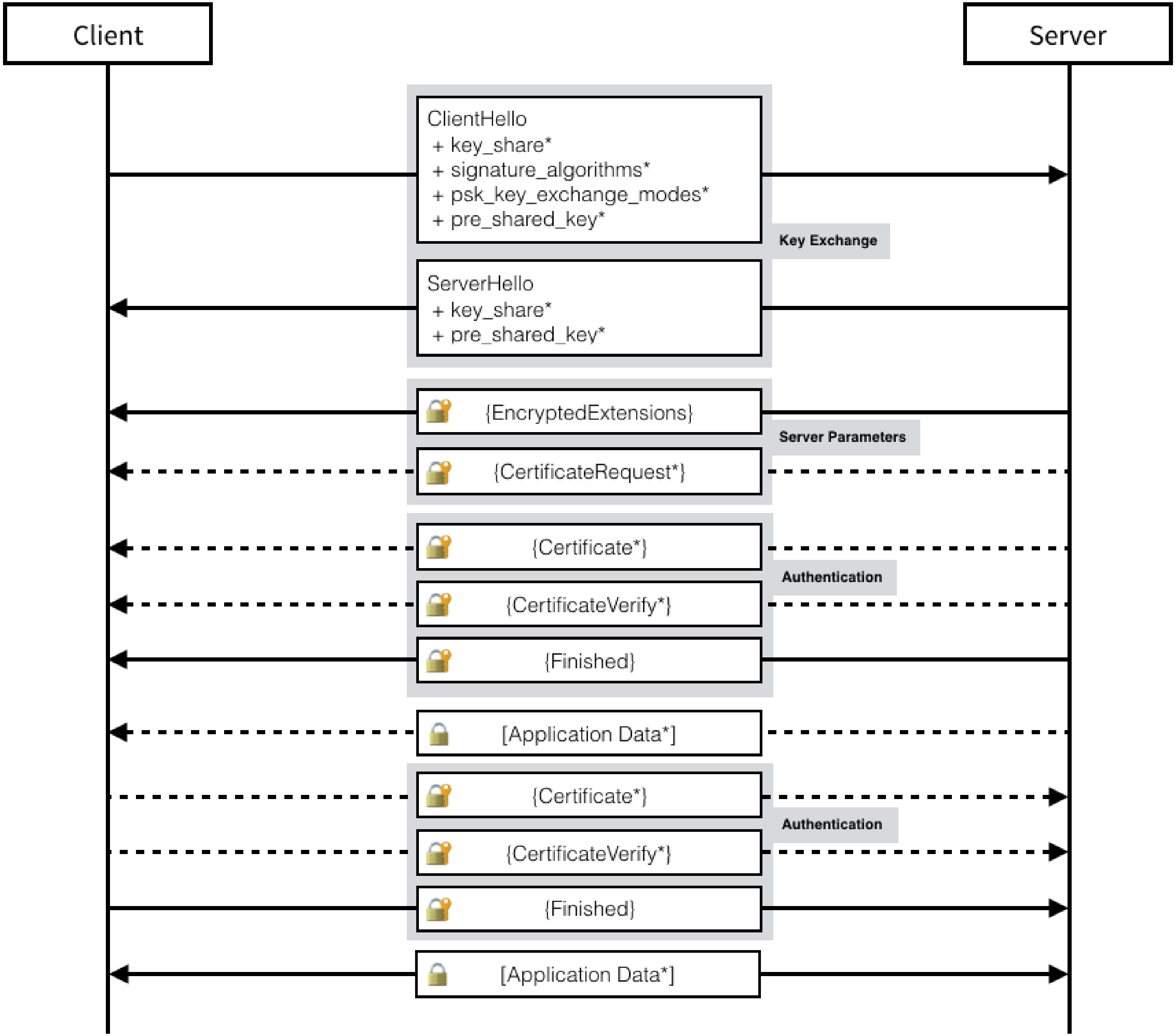

Client
Server
Client Hello
Server Hello
Init
Server's Certificate, ServerKeyExchange,Certificate
Request(all optional)
Server Hello Done
Hello Message Phase
Client's Certificate
(Optional, if the server requested for it)
ClientKeyExchange
(Depends on PK algorithm selected during Hello EX)
Certificate Verify
(optional:, signed certificate sent to verify clients certificate)
Client Exchange
Change CipherSpec
(Signals transition in ciphering methods)
Finished
Change CIpherSpec
Finished
Cipher Change

Handshake of a Reused Session





```
192.168.1.3       217.169.192.73    TLSv1   Client Hello
217.169.192.73    192.168.1.3       TLSv1   Server Hello, Change Cipher Spec
217.169.192.73    192.168.1.3       TLSv1   Encrypted Handshake Message
192.168.1.3       217.169.192.73    TCP     35451 > https [ACK] Seq=167 Ack=123 Win=16438 Len=0
192.168.1.3       217.169.192.73    TLSv1   Change Cipher Spec, Encrypted Handshake Message
192.168.1.3       217.169.192.73    TLSv1   Application Data
```



Client

Server

ClientHello
+ key_share*
+ signature_algorithms*
+ psk_key_exchange_modes*
+ pre_shared_key*

Key Exchange

ServerHello
+ key_share*
+ pre_shared_key*

{EncryptedExtensions}

Server Parameters

{CertificateRequest*}

{Certificate*}

Authentication

{CertificateVerify*}

{Finished}

[Application Data*]

{Certificate*}

Authentication

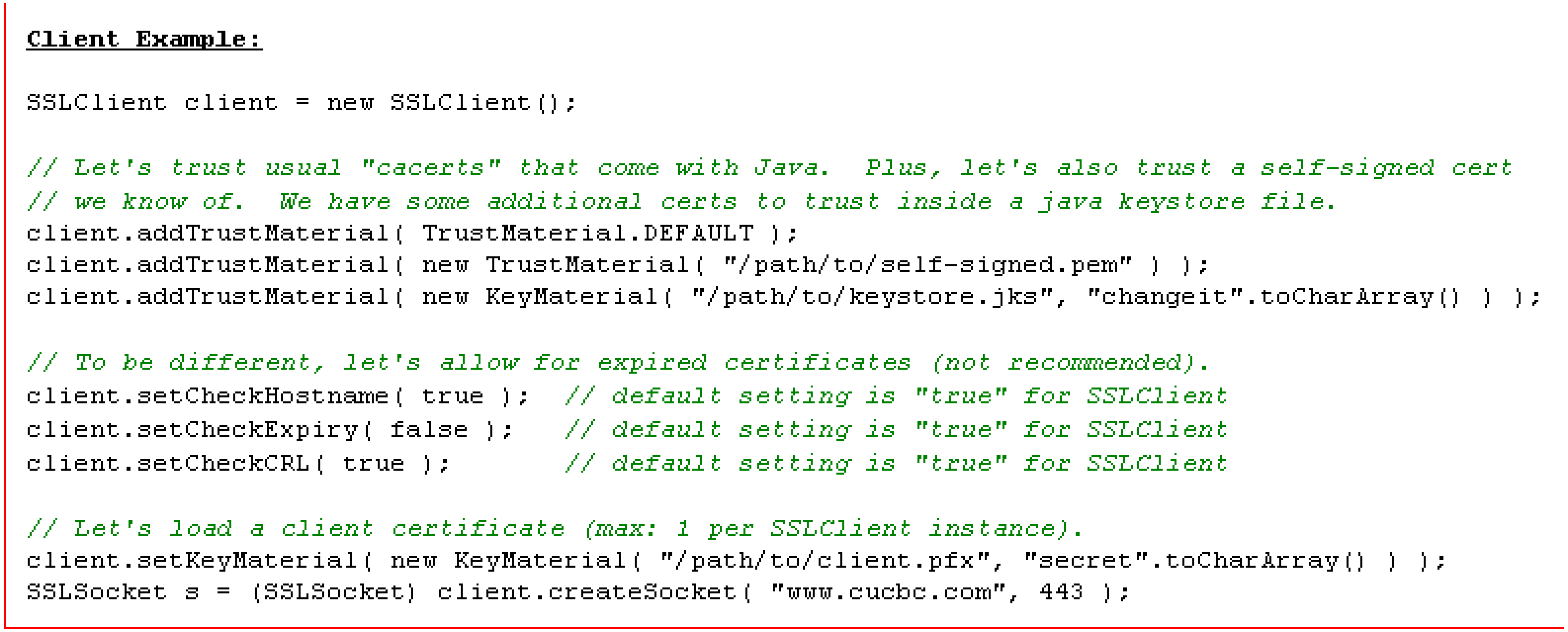
{CertificateVerify*}

{Finished}

[Application Data*]

Seguridad en IP: TLS

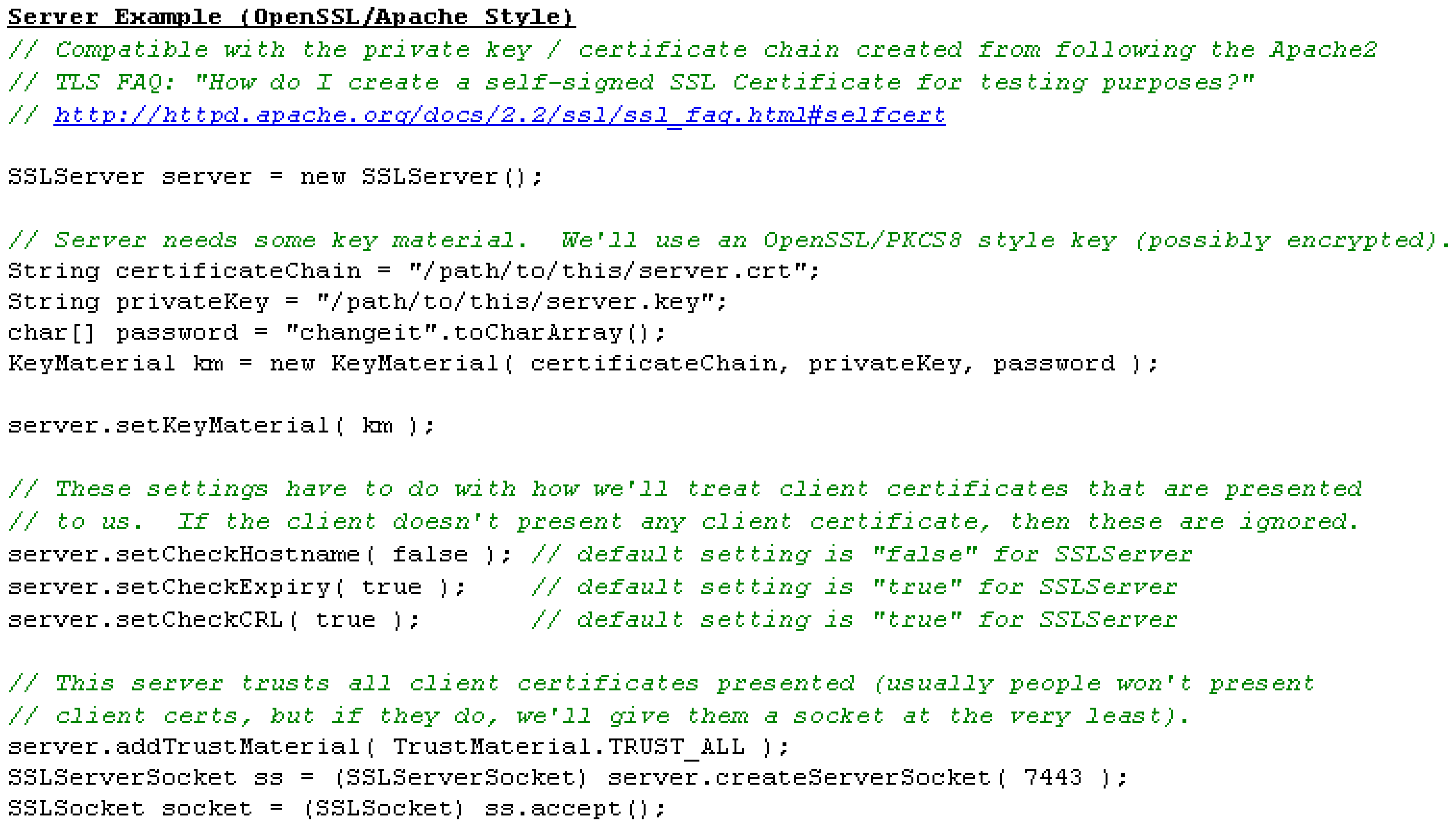
TLS APIs



Client Example:

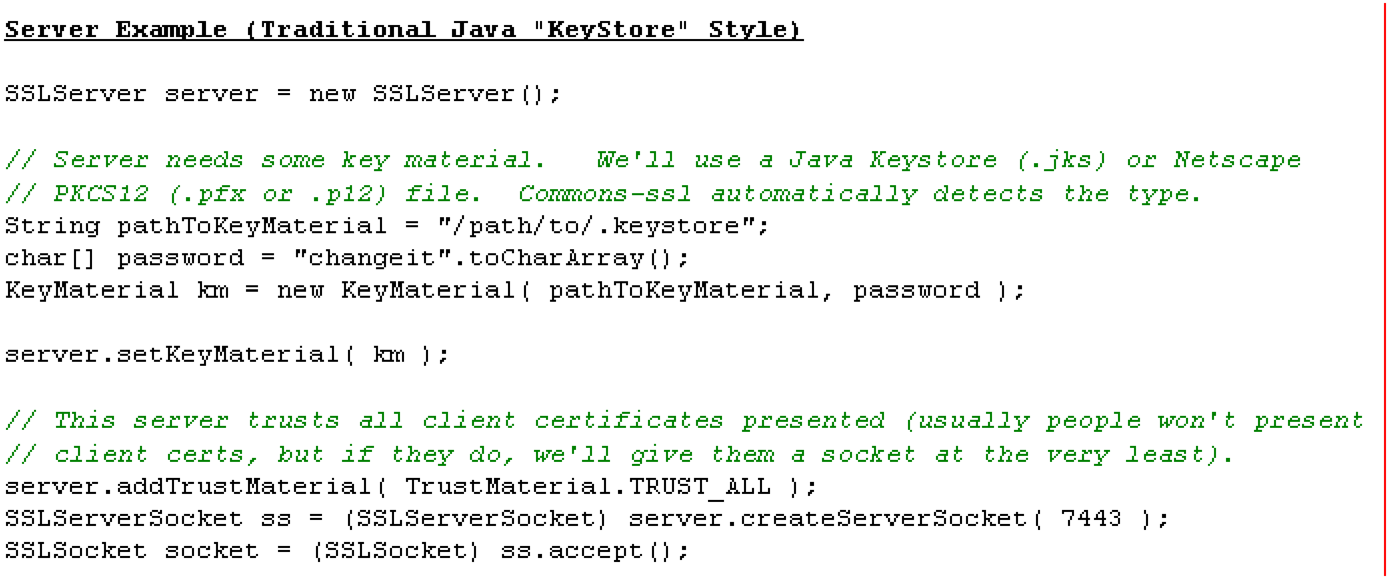
```
SSLClient client = new SSLClient();

// Let's trust usual "cacerts" that come with Java.  Plus, let's also trust a self-signed cert
// we know of.  We have some additional certs to trust inside a java keystore file.
client.addTrustMaterial( TrustMaterial.DEFAULT );
client.addTrustMaterial( new TrustMaterial( "/path/to/self-signed.pem" ) );
client.addTrustMaterial( new KeyMaterial( "/path/to/keystore.jks", "changeit".toCharArray() ) );

// To be different, let's allow for expired certificates (not recommended).
client.setCheckHostname( true );  // default setting is "true" for SSLClient
client.setCheckExpiry( false );   // default setting is "true" for SSLClient
client.setCheckCRL( true );       // default setting is "true" for SSLClient

// Let's load a client certificate (max: 1 per SSLClient instance).
client.setKeyMaterial( new KeyMaterial( "/path/to/client.pfx", "secret".toCharArray() ) );
SSLSocket s = (SSLSocket) client.createSocket( "www.cucbc.com", 443 );
```

https://stackoverflow.com/questions/49866171/how-to-use-apache-commons-sslserver-sslclient-with-certificate



Server Example (OpenSSL/Apache Style)

```
// Compatible with the private key / certificate chain created from following the Apache2
// TLS FAQ: "How do I create a self-signed SSL Certificate for testing purposes?"
// http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html#selfcert

SSLServer server = new SSLServer();

// Server needs some key material.  We'll use an OpenSSL/PKCS8 style key (possibly encrypted).
String certificateChain = "/path/to/this/server.crt";
String privateKey = "/path/to/this/server.key";
char[] password = "changeit".toCharArray();
KeyMaterial km = new KeyMaterial( certificateChain, privateKey, password );

server.setKeyMaterial( km );

// These settings have to do with how we'll treat client certificates that are presented
// to us.  If the client doesn't present any client certificate, then these are ignored.
server.setCheckHostname( false ); // default setting is "false" for SSLServer
server.setCheckExpiry( true );    // default setting is "true" for SSLServer
server.setCheckCRL( true );       // default setting is "true" for SSLServer

// This server trusts all client certificates presented (usually people won't present
// client certs, but if they do, we'll give them a socket at the very least).
server.addTrustMaterial( TrustMaterial.TRUST_ALL );
SSLServerSocket ss = (SSLServerSocket) server.createServerSocket( 7443 );
SSLSocket socket = (SSLSocket) ss.accept();
```



Server Example (Traditional Java "KeyStore" Style)

```
SSLServer server = new SSLServer();

// Server needs some key material.   We'll use a Java Keystore (.jks) or Netscape
// PKCS12 (.pfx or .p12) file.  Commons-ssl automatically detects the type.
String pathToKeyMaterial = "/path/to/.keystore";
char[] password = "changeit".toCharArray();
KeyMaterial km = new KeyMaterial( pathToKeyMaterial, password );

server.setKeyMaterial( km );

// This server trusts all client certificates presented (usually people won't present
// client certs, but if they do, we'll give them a socket at the very least).
server.addTrustMaterial( TrustMaterial.TRUST_ALL );
SSLServerSocket ss = (SSLServerSocket) server.createServerSocket( 7443 );
SSLSocket socket = (SSLSocket) ss.accept();
```

Seguridad en IP: TLS

stunnel

STunnel: POP3 server example

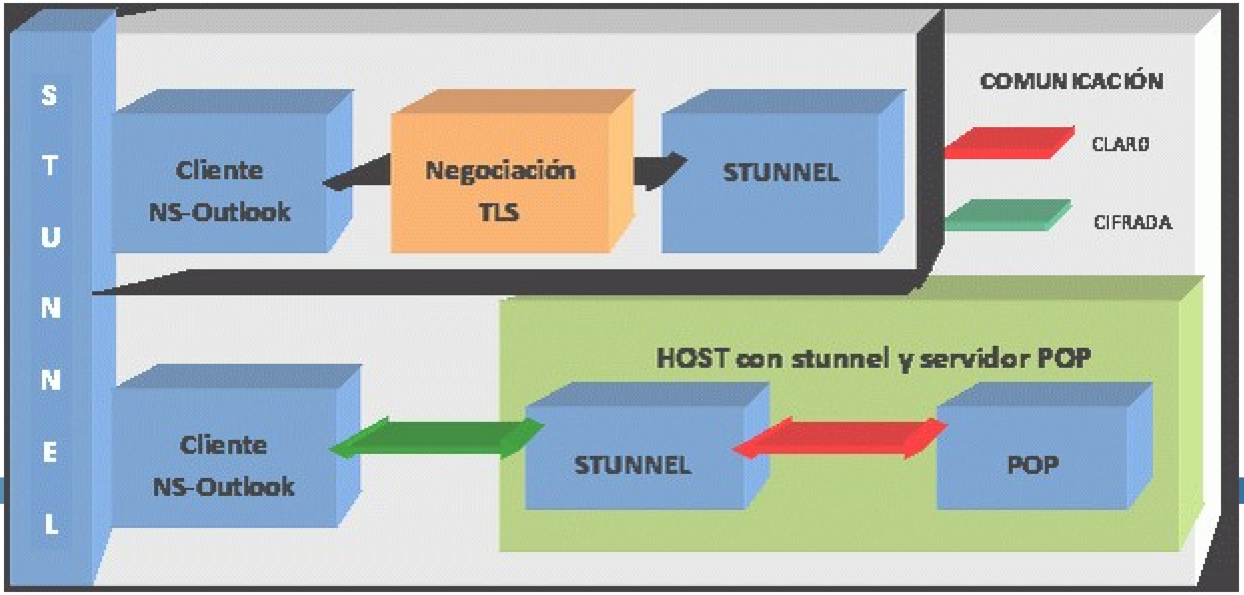
www.stunnel.org

Es un proxy que agrega SSL

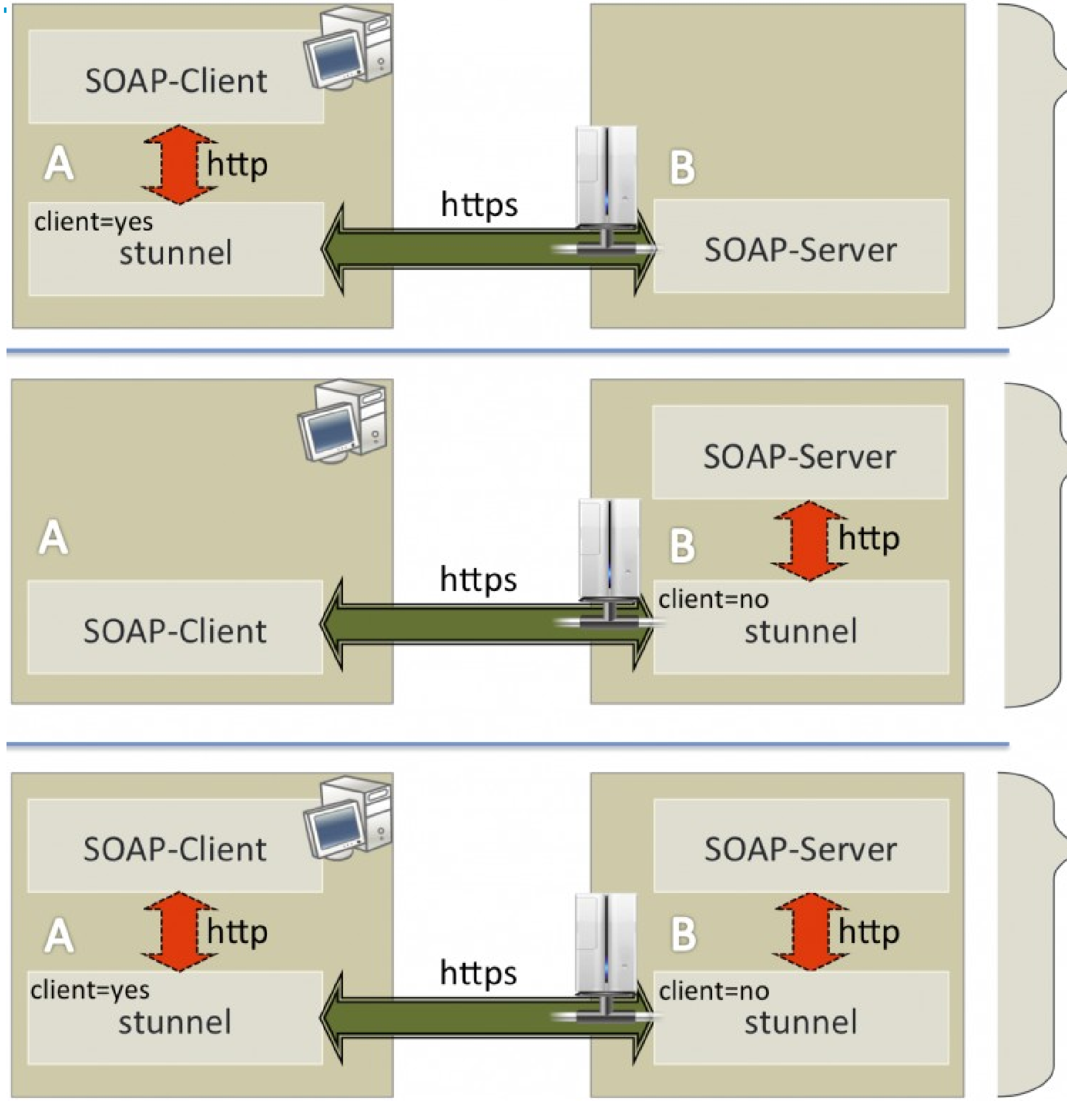
a cualquier protocolo

pop3s (995):

```
/usr/sbin/stunnel -r localhost:pop3 -p /path/to/stunnel.pem
```



Ejemplo de stunnel con SOAP



If
Server exports https
Client doesn't support https

stunnel has client mode “yes"

If
Server export http
Client wants to talk https

stunnel has client mode “no"

If
Server and **client** don't use https (and we need to make Secure communication)

stunnel has client mode “yes" on **A** and “no” on server **B**.

https://busylog.net/stunnel-how-to-shield-a-protocol/

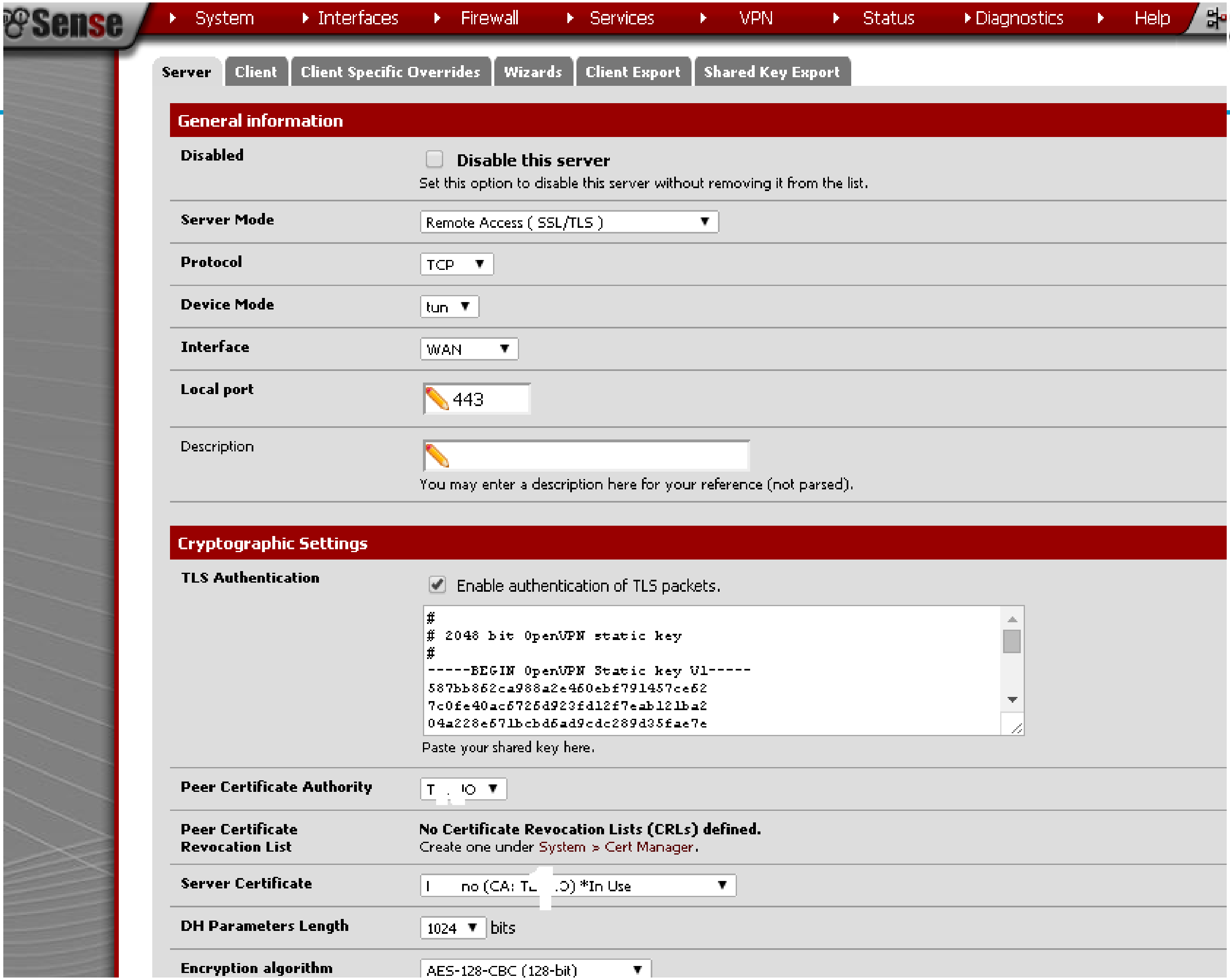
Seguridad en IP: TLS

Openvpn

```
# Tunnel mode
dev tun
mode server

# Server endpoint appears first, followed by the gateway interface ip
ifconfig 10.1.0.1 10.1.0.2
# Range of IP addresses reserved for clients
ifconfig-pool 10.1.0.4 10.1.0.254
route 10.1.0.0 255.255.255.0

push "route 10.1.0.1 255.255.255.255"

# Specify tls-server for certificate exchange
tls-server

dh dh1024.pem
# Root certificate
ca CA-DB/cacert.pem
cert vpncert.pem
key vpnkey.pem

# Check for revoked client certificates.
crl-verify CA-DB/crl/crl.pem
```

Configuracion cliente

```
#OpenVPN Server conf
tls-client
client
dev tun
proto tcp
remote 200.88.88.89 443
pkcs12 certificado.p12
cipher AES-CBC
verb 3
ns-cert-type server
```



General information

Disabled	□ **Disable this server** Set this option to disable this server without removing it from the list.
Server Mode	Remote Access (SSL/TLS) ▼
Protocol	TCP ▼
Device Mode	tun ▼
Interface	WAN ▼
Local port	443
Description	 You may enter a description here for your reference (not parsed).

Cryptographic Settings

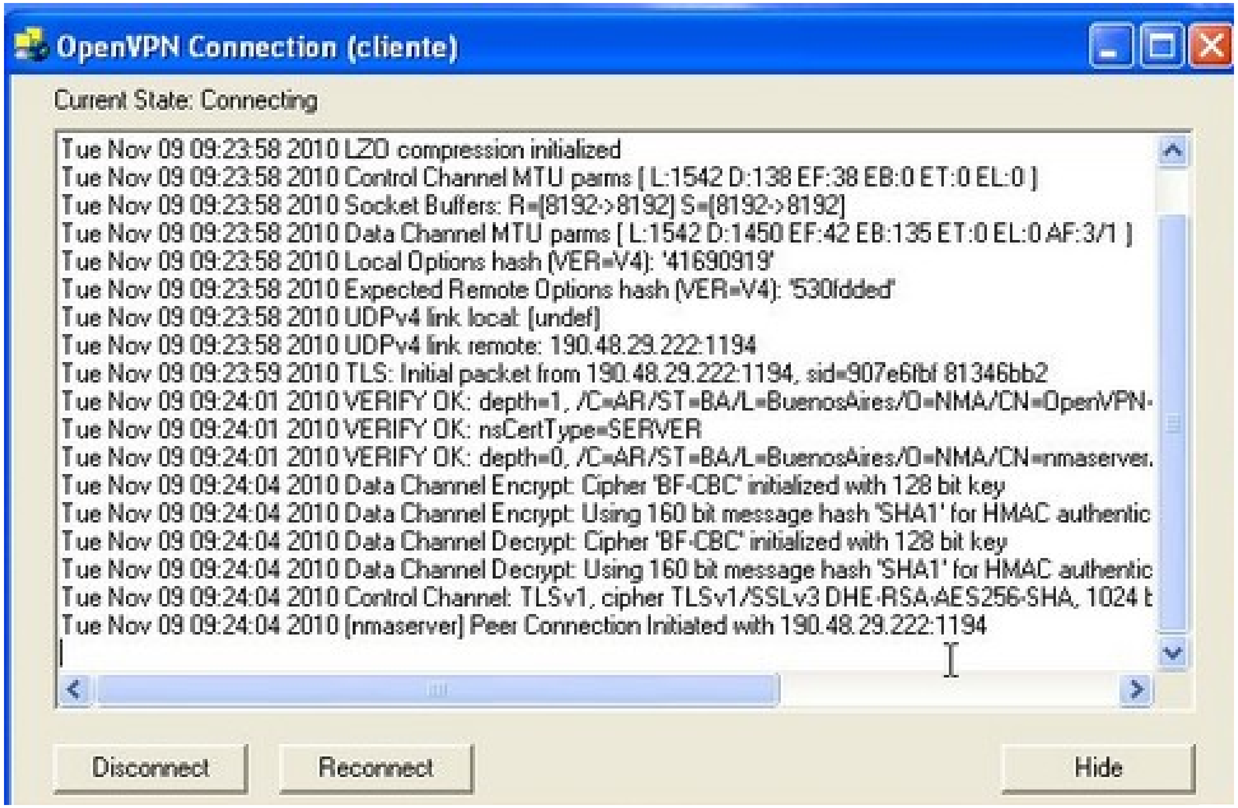
TLS Authentication

☑ Enable authentication of TLS packets.

```
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
587bb862ca988a2e460ebf791457ce62
7c0fe40ac6725d923fd12f7eab121ba2
04a228e671bcbd6ad9cdc289d35fae7e
```

Paste your shared key here.

Peer Certificate Authority	T . !O ▼
Peer Certificate Revocation List	**No Certificate Revocation Lists (CRLs) defined.** Create one under System > Cert Manager.
Server Certificate	I no (CA: TL .O) *In Use ▼
DH Parameters Length	1024 ▼ bits
Encryption algorithm	AES-128-CBC (128-bit) ▼



Current State: Connecting

```
Tue Nov 09 09:23:58 2010 LZO compression initialized
Tue Nov 09 09:23:58 2010 Control Channel MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Tue Nov 09 09:23:58 2010 Socket Buffers: R=[8192->8192] S=[8192->8192]
Tue Nov 09 09:23:58 2010 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:135 ET:0 EL:0 AF:3/1 ]
Tue Nov 09 09:23:58 2010 Local Options hash (VER=V4): '41690919'
Tue Nov 09 09:23:58 2010 Expected Remote Options hash (VER=V4): '530fdded'
Tue Nov 09 09:23:58 2010 UDPv4 link local: [undef]
Tue Nov 09 09:23:58 2010 UDPv4 link remote: 190.48.29.222:1194
Tue Nov 09 09:23:59 2010 TLS: Initial packet from 190.48.29.222:1194, sid=907e6fbf 81346bb2
Tue Nov 09 09:24:01 2010 VERIFY OK: depth=1, /C=AR/ST=BA/L=BuenosAires/O=NMA/CN=OpenVPN-
Tue Nov 09 09:24:01 2010 VERIFY OK: nsCertType=SERVER
Tue Nov 09 09:24:01 2010 VERIFY OK: depth=0, /C=AR/ST=BA/L=BuenosAires/O=NMA/CN=nmaserver.
Tue Nov 09 09:24:04 2010 Data Channel Encrypt: Cipher 'BF-CBC' initialized with 128 bit key
Tue Nov 09 09:24:04 2010 Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC authentic
Tue Nov 09 09:24:04 2010 Data Channel Decrypt: Cipher 'BF-CBC' initialized with 128 bit key
Tue Nov 09 09:24:04 2010 Data Channel Decrypt: Using 160 bit message hash 'SHA1' for HMAC authentic
Tue Nov 09 09:24:04 2010 Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 1024 b
Tue Nov 09 09:24:04 2010 [nmaserver] Peer Connection Initiated with 190.48.29.222:1194
```

```
>ipconfig

Configuración IP de Windows
Adaptador de Ethernet Conexión de área local 3:

   Sufijo DNS específico para la conexión. . :
   Dirección IPv4. . . . . . . . . . . . . . : 10.22.22.2
   Máscara de subred . . . . .. : 255.255.255.252
   Puerta de enlace predeterminada . . . . . :
```

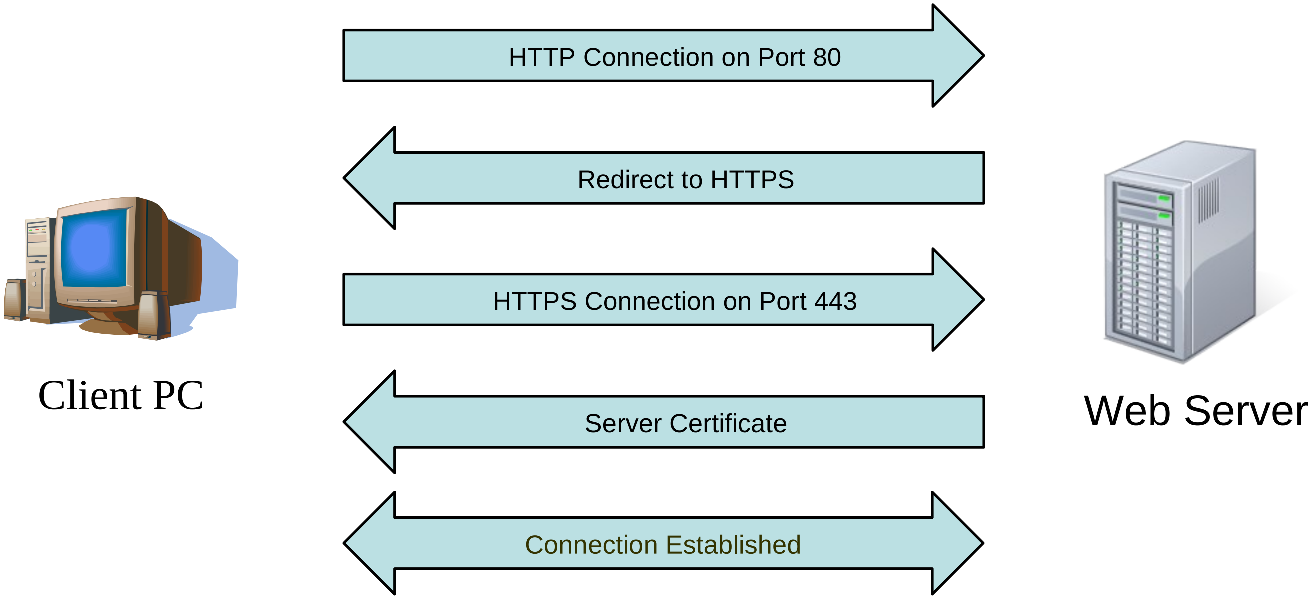
- Embebido en sistemas de facil instalacion como pfsense o ipcop.org
- Casi todas las redes permiten SSL por sus proxys
  - Difícil de diferenciar de una conexión SSL a una pagina web cualquiera
  - Permite salir de ambientes intranet

  .

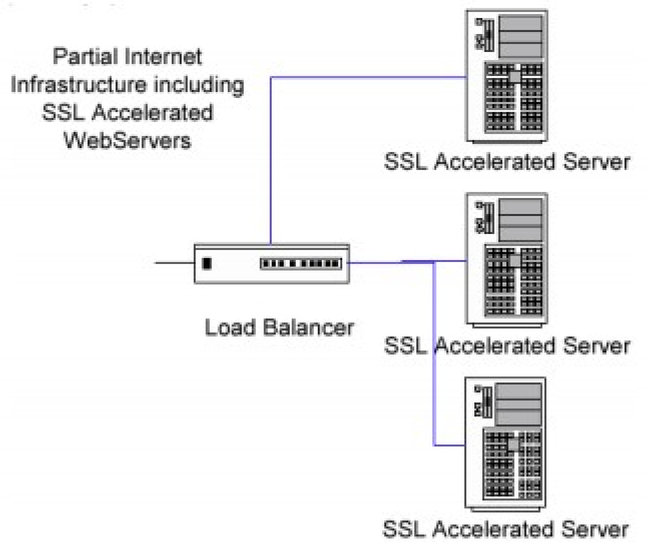
Seguridad en IP: TLS

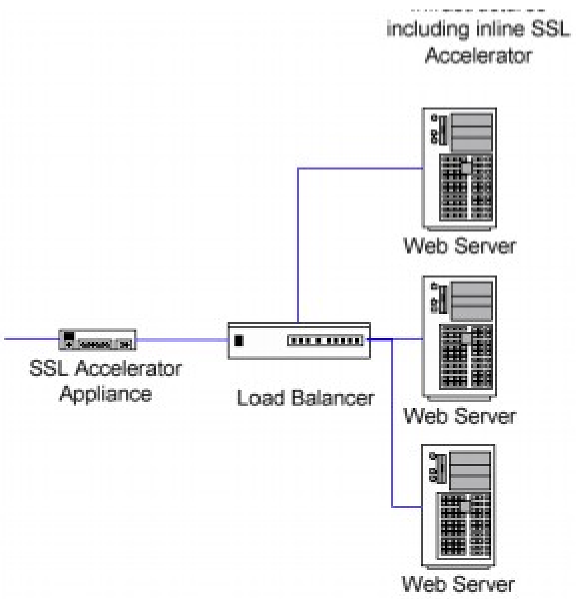
Terminador SSL

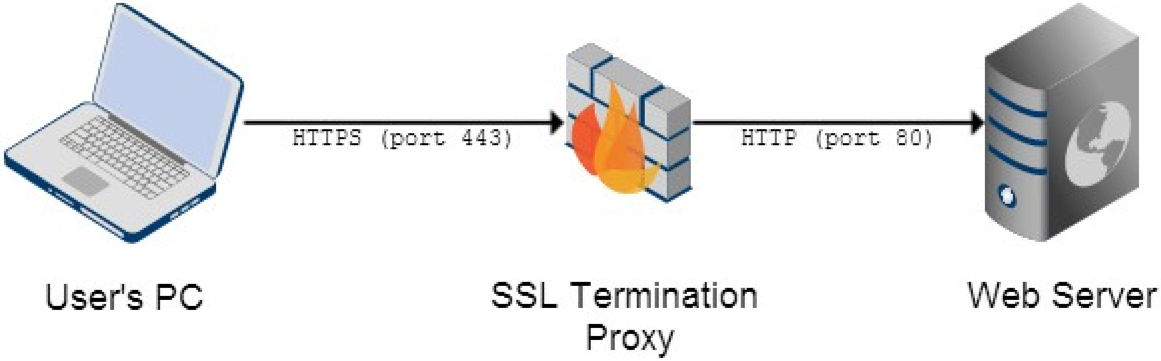
Conexión Tipica

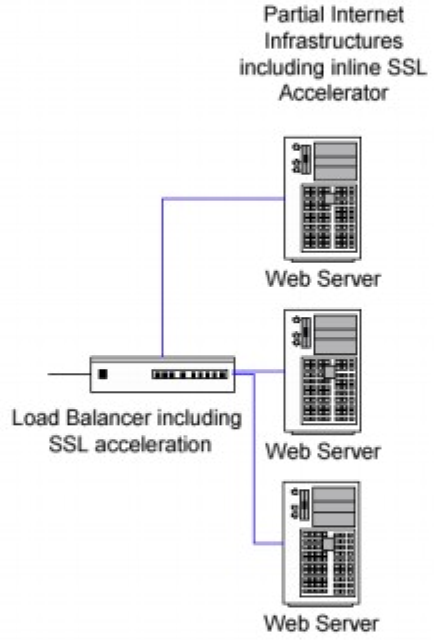


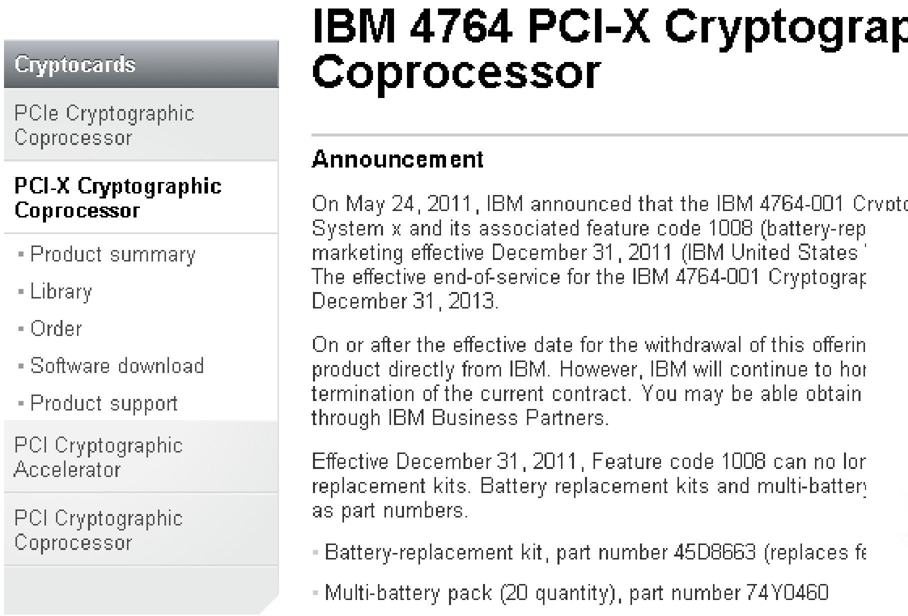
Terminadores SSL








Partial Internet Infrastructures including inline SSL Accelerator
Web Server
Web Server
Load Balancer including SSL acceleration
Web Server



Cryptocards

PCIe Cryptographic Coprocessor

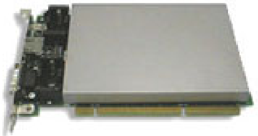
PCI-X Cryptographic Coprocessor

- Product summary
- Library
- Order
- Software download
- Product support

PCI Cryptographic Accelerator

PCI Cryptographic Coprocessor

IBM 4764 PCI-X Cryptographic Coprocessor



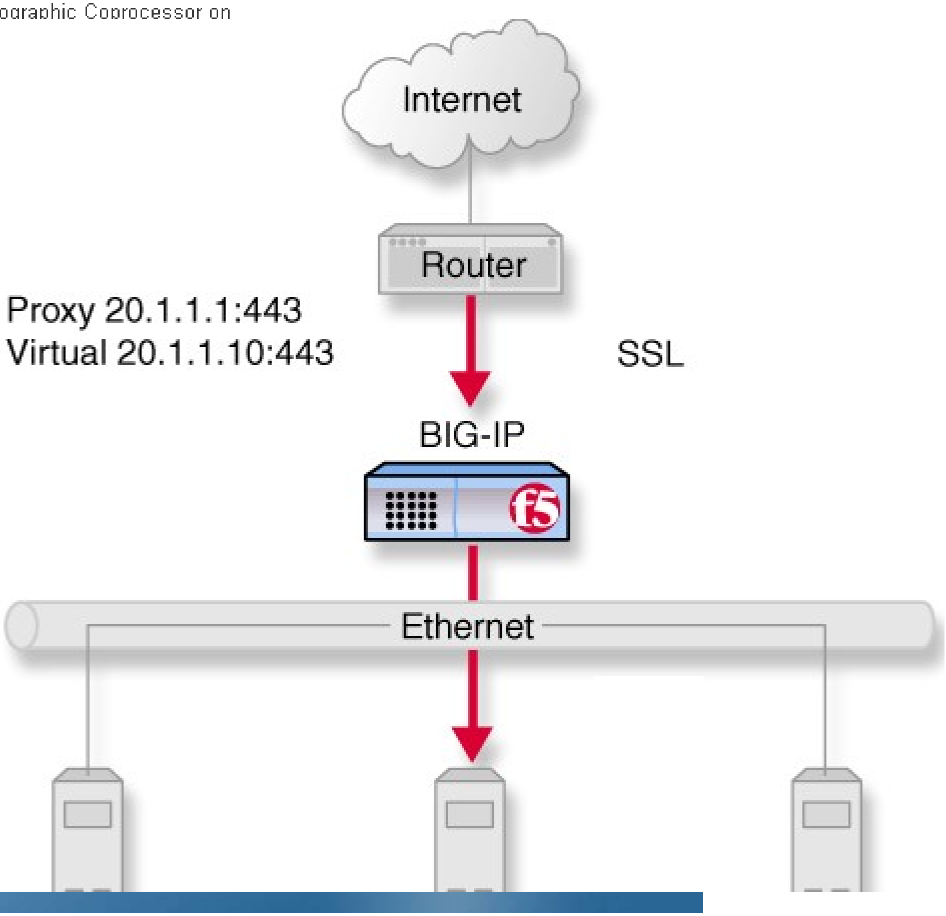
Announcement

On May 24, 2011, IBM announced that the IBM 4764-001 Cryptographic Coprocessor on System x and its associated feature code 1008 (battery-rep marketing effective December 31, 2011 (IBM United States The effective end-of-service for the IBM 4764-001 Cryptograp December 31, 2013.

On or after the effective date for the withdrawal of this offerin product directly from IBM. However, IBM will continue to hon termination of the current contract. You may be able obtain through IBM Business Partners.

Effective December 31, 2011, Feature code 1008 can no lor replacement kits. Battery replacement kits and multi-battery as part numbers.

- Battery-replacement kit, part number 45D8663 (replaces fe
- Multi-battery pack (20 quantity), part number 74Y0460



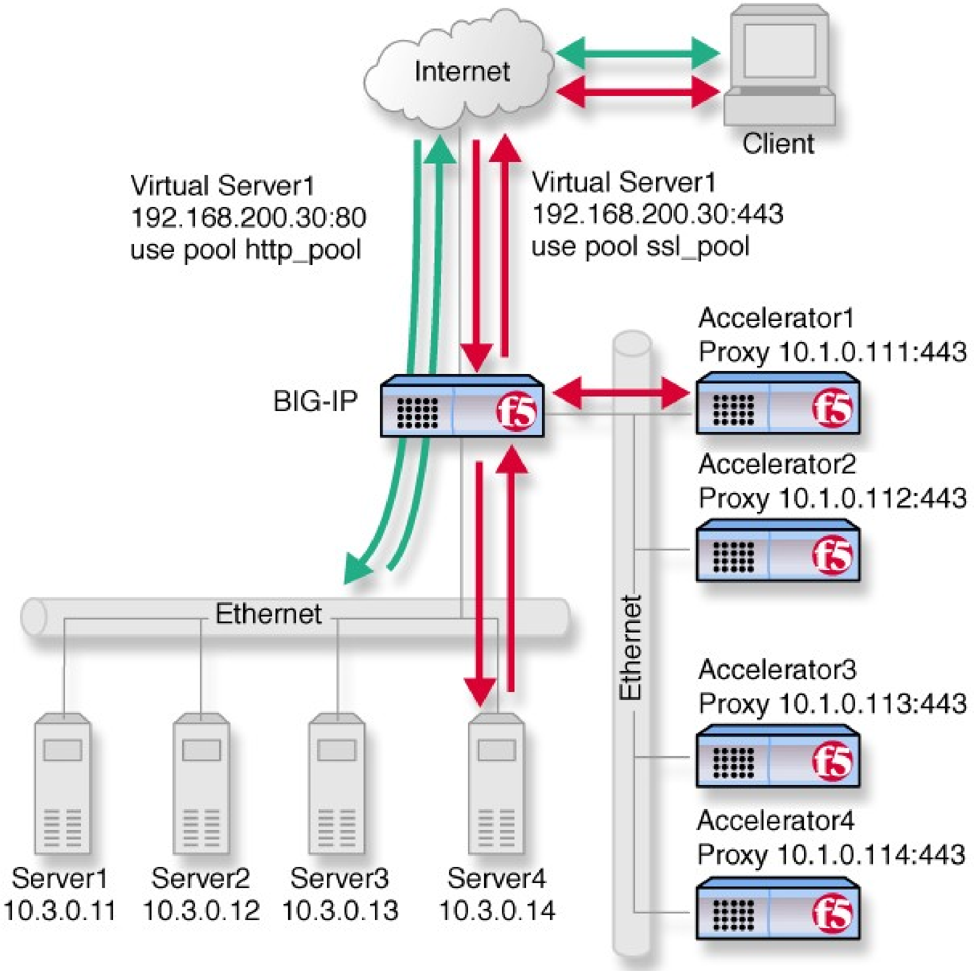
Proxy 20.1.1.1:443
Virtual 20.1.1.10:443

SSL

INTERACTUANDO CON SSL/TLS

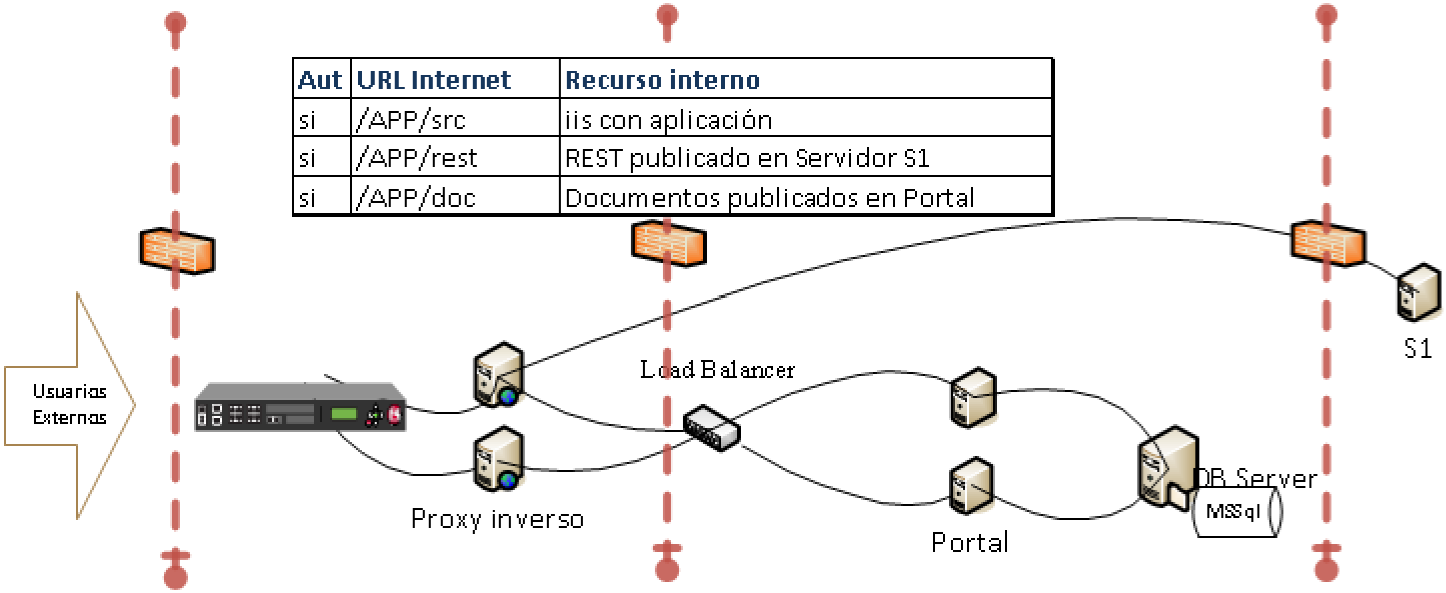
Aceleradores SSL

- Permiten delegar el trabajo de computo requerido por el Handshake en el uso de las claves asincrónicas, permitiendo que un hardware especializado realice la tarea.



Fuente: http://support.f5.com/kb/en-us/archived_products/big-ip/manuals/product/bigip4_5_10ecommerce/bigip_sslgate.html

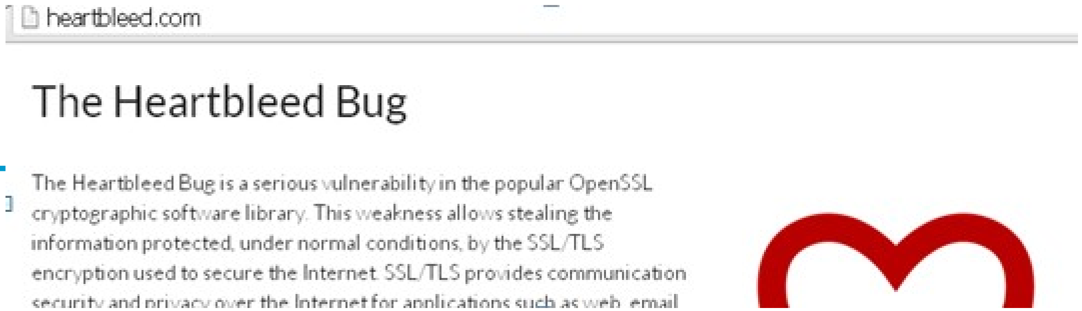
Donde poner SSL?



Aut	URL Internet	Recurso interno
si	/APP/src	iis con aplicación
si	/APP/rest	REST publicado en Servidor S1
si	/APP/doc	Documentos publicados en Portal

Seguridad en IP: TLS

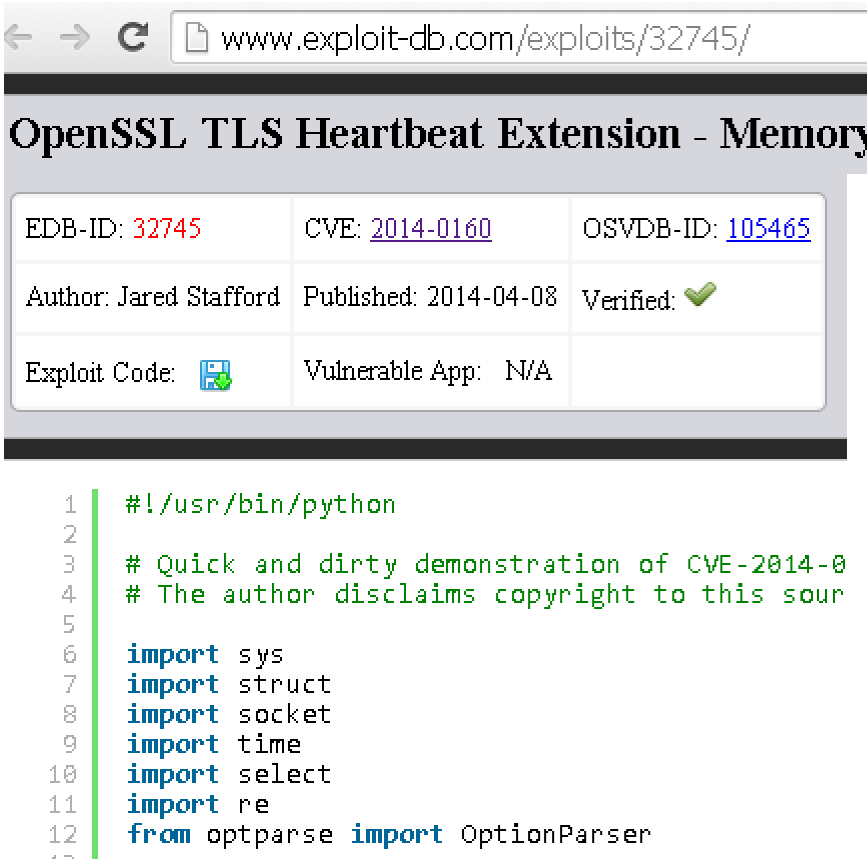
Bugs



heartbleed.com

The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email

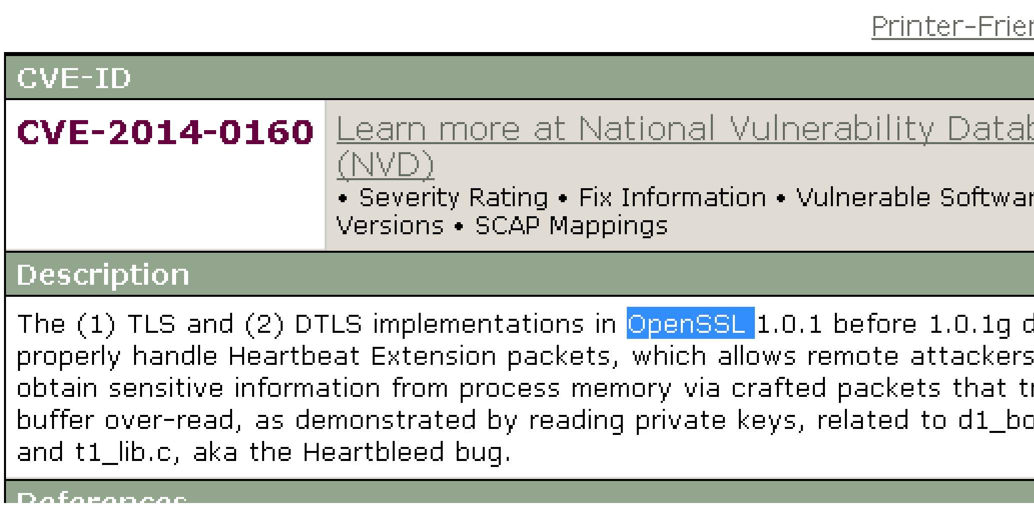


www.exploit-db.com/exploits/32745/

OpenSSL TLS Heartbeat Extension - Memory Disclosure

EDB-ID: 32745	CVE: 2014-0160	OSVDB-ID: 105465
Author: Jared Stafford	Published: 2014-04-08	Verified: ✓
Exploit Code:	Vulnerable App: N/A	

```
#!/usr/bin/python

# Quick and dirty demonstration of CVE-2014-0
# The author disclaims copyright to this sour

import sys
import struct
import socket
import time
import select
import re
from optparse import OptionParser
```



Printer-Frier

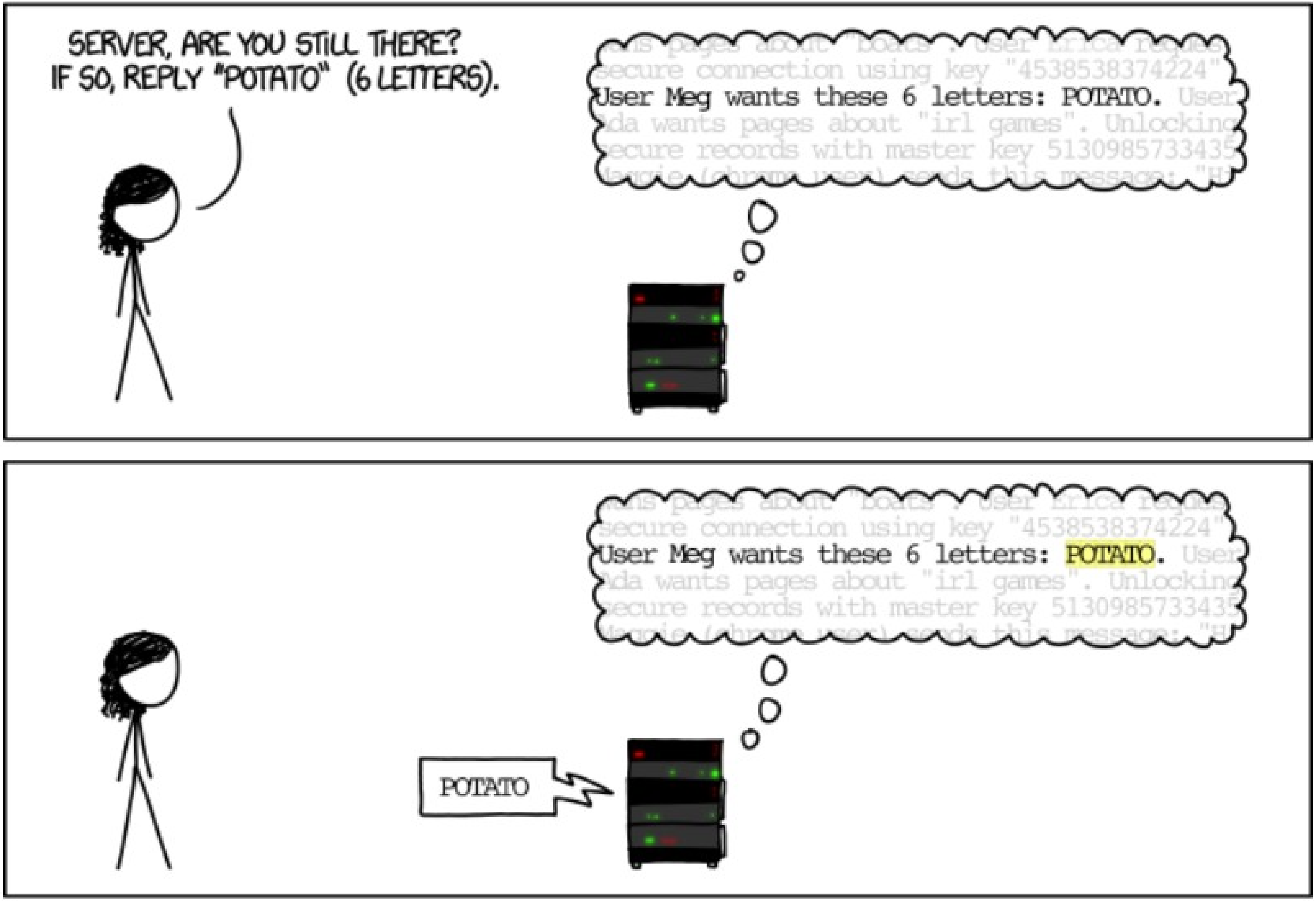
CVE-ID	
CVE-2014-0160	Learn more at National Vulnerability Datab (NVD) • Severity Rating • Fix Information • Vulnerable Softwar Versions • SCAP Mappings

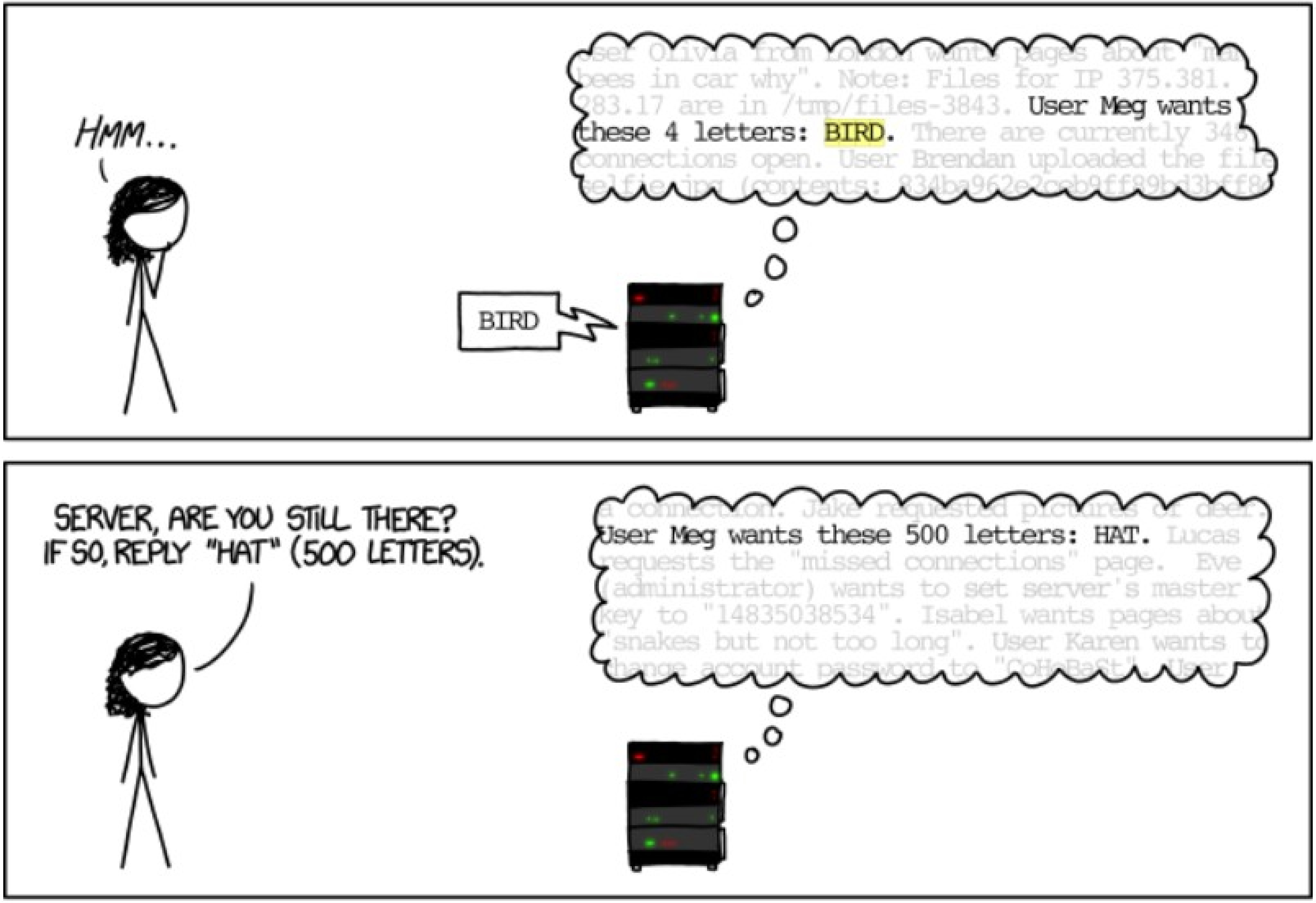
Description

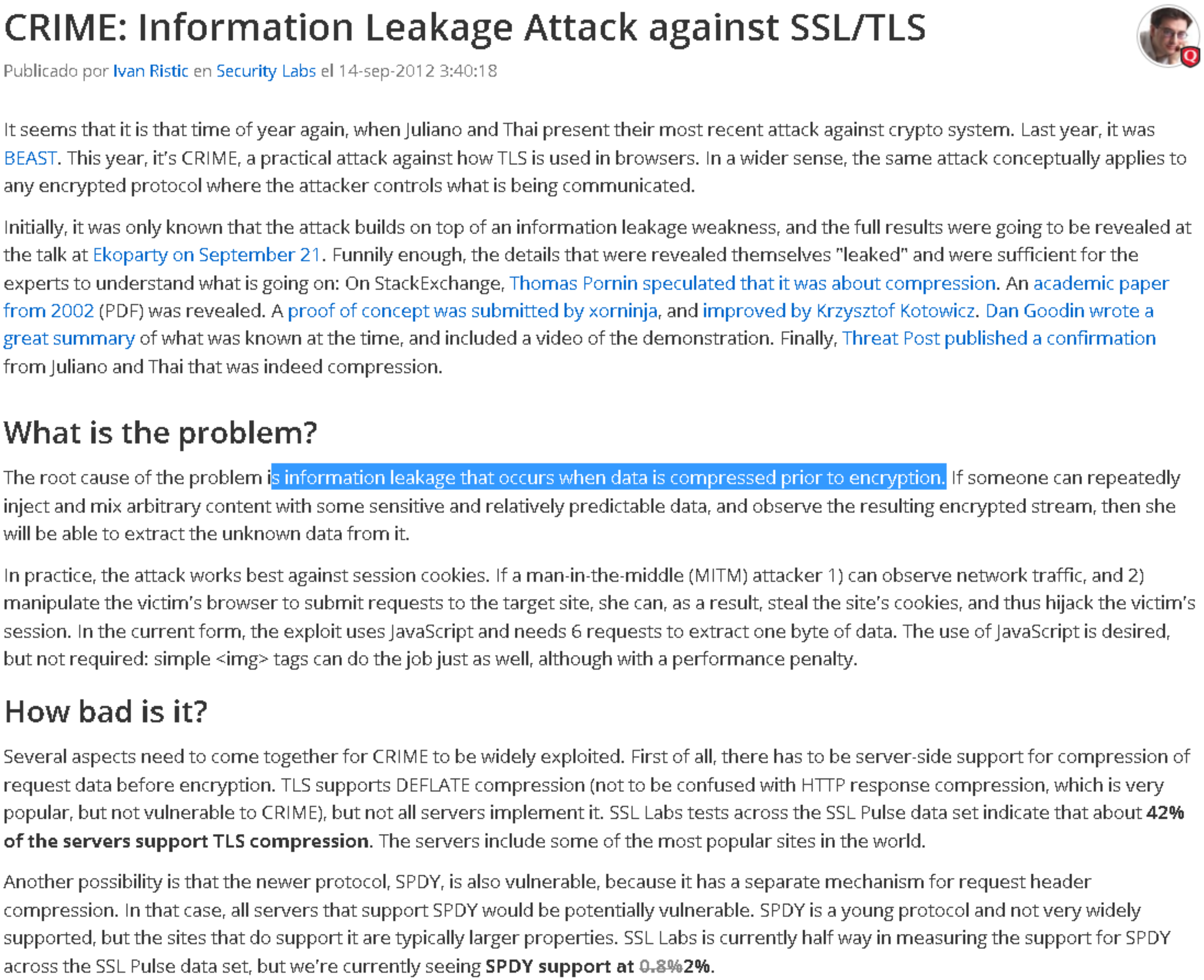
The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g d properly handle Heartbeat Extension packets, which allows remote attackers obtain sensitive information from process memory via crafted packets that tr buffer over-read, as demonstrated by reading private keys, related to d1_bo and t1_lib.c, aka the Heartbleed bug.

References

HOW THE HEARTBLEED BUG WORKS:


SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
ants pages about "boats". User Erica reques
secure connection using key "4538538374224"
User Meg wants these 6 letters: POTATO. User
Ada wants pages about "irl games". Unlocking
secure records with master key 5130985733435
Meggie (chrome user) sends this message: "H
POTATO


HMM...
User Olivia from London wants pages about "ma bees in car why". Note: Files for IP 375.381. 283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 348 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User



CRIME: Information Leakage Attack against SSL/TLS

Publicado por Ivan Ristic en Security Labs el 14-sep-2012 3:40:18

It seems that it is that time of year again, when Juliano and Thai present their most recent attack against crypto system. Last year, it was BEAST. This year, it's CRIME, a practical attack against how TLS is used in browsers. In a wider sense, the same attack conceptually applies to any encrypted protocol where the attacker controls what is being communicated.

Initially, it was only known that the attack builds on top of an information leakage weakness, and the full results were going to be revealed at the talk at Ekoparty on September 21. Funnily enough, the details that were revealed themselves "leaked" and were sufficient for the experts to understand what is going on: On StackExchange, Thomas Pornin speculated that it was about compression. An academic paper from 2002 (PDF) was revealed. A proof of concept was submitted by xorninja, and improved by Krzysztof Kotowicz. Dan Goodin wrote a great summary of what was known at the time, and included a video of the demonstration. Finally, Threat Post published a confirmation from Juliano and Thai that was indeed compression.

What is the problem?

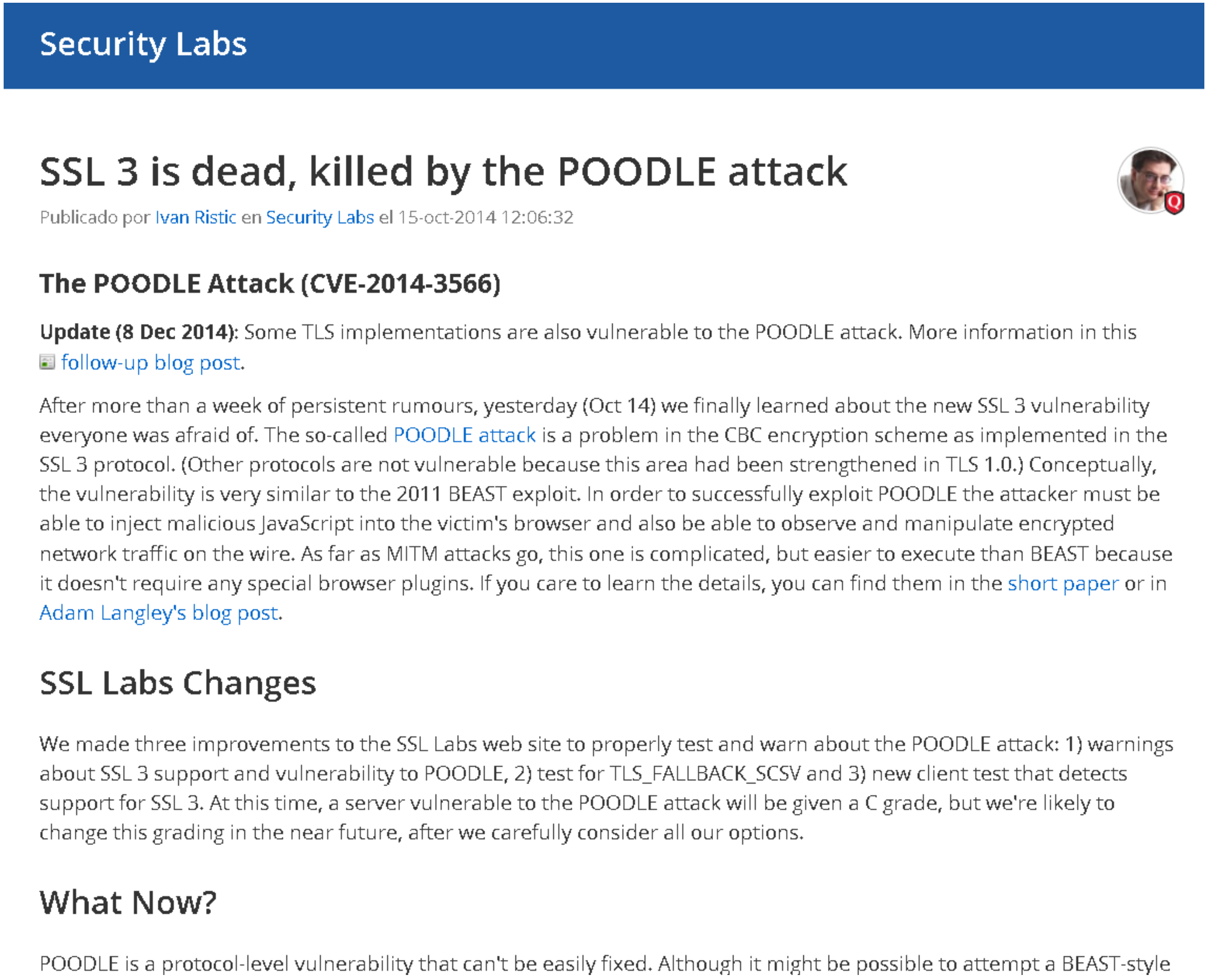
The root cause of the problem is information leakage that occurs when data is compressed prior to encryption. If someone can repeatedly inject and mix arbitrary content with some sensitive and relatively predictable data, and observe the resulting encrypted stream, then she will be able to extract the unknown data from it.

In practice, the attack works best against session cookies. If a man-in-the-middle (MITM) attacker 1) can observe network traffic, and 2) manipulate the victim's browser to submit requests to the target site, she can, as a result, steal the site's cookies, and thus hijack the victim's session. In the current form, the exploit uses JavaScript and needs 6 requests to extract one byte of data. The use of JavaScript is desired, but not required: simple <img> tags can do the job just as well, although with a performance penalty.

How bad is it?

Several aspects need to come together for CRIME to be widely exploited. First of all, there has to be server-side support for compression of request data before encryption. TLS supports DEFLATE compression (not to be confused with HTTP response compression, which is very popular, but not vulnerable to CRIME), but not all servers implement it. SSL Labs tests across the SSL Pulse data set indicate that about **42% of the servers support TLS compression**. The servers include some of the most popular sites in the world.

Another possibility is that the newer protocol, SPDY, is also vulnerable, because it has a separate mechanism for request header compression. In that case, all servers that support SPDY would be potentially vulnerable. SPDY is a young protocol and not very widely supported, but the sites that do support it are typically larger properties. SSL Labs is currently half way in measuring the support for SPDY across the SSL Pulse data set, but we're currently seeing **SPDY support at ~~0.8%~~2%**.



Security Labs

SSL 3 is dead, killed by the POODLE attack

Publicado por Ivan Ristic en Security Labs el 15-oct-2014 12:06:32

The POODLE Attack (CVE-2014-3566)

Update (8 Dec 2014): Some TLS implementations are also vulnerable to the POODLE attack. More information in this follow-up blog post.

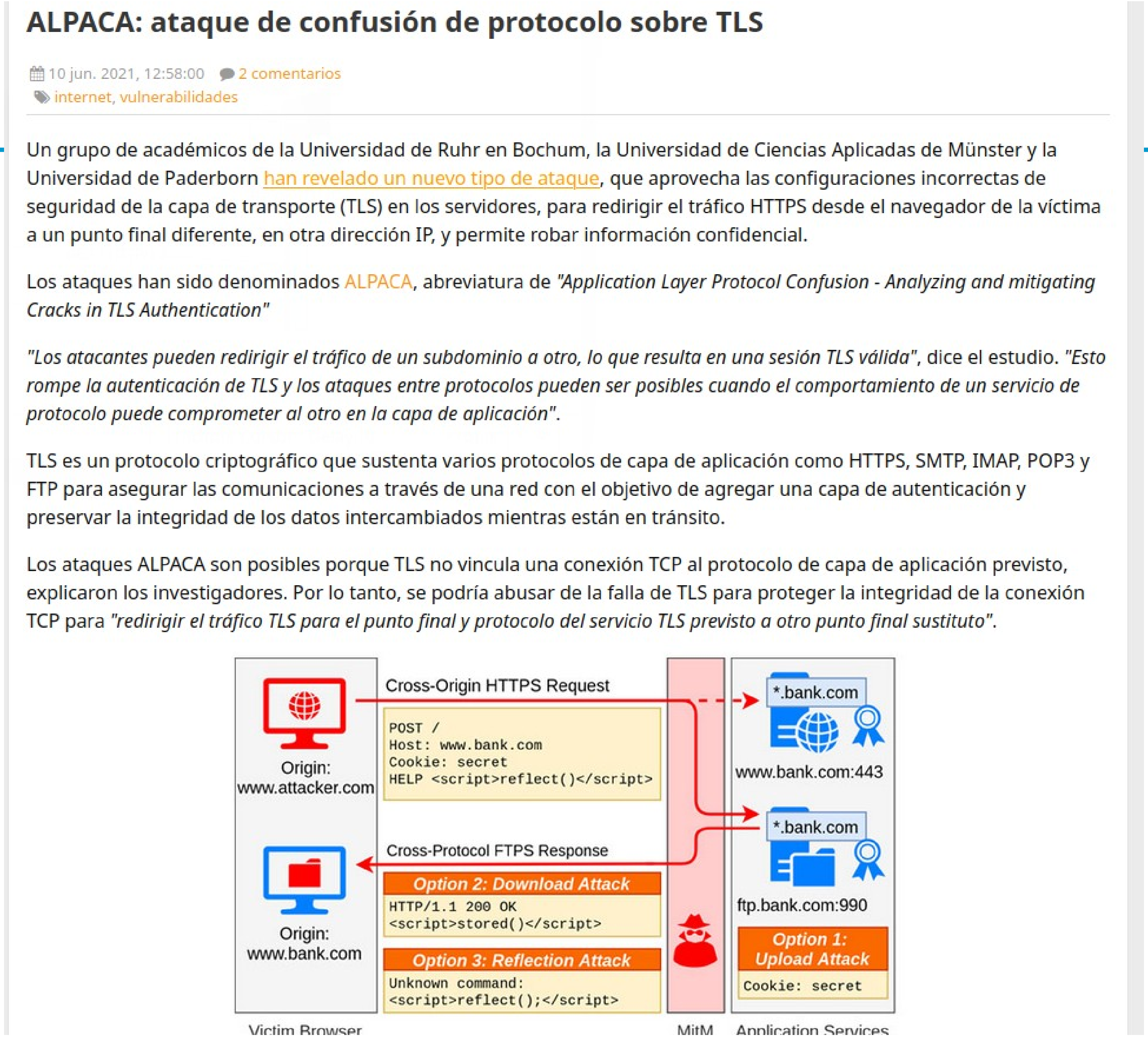
After more than a week of persistent rumours, yesterday (Oct 14) we finally learned about the new SSL 3 vulnerability everyone was afraid of. The so-called POODLE attack is a problem in the CBC encryption scheme as implemented in the SSL 3 protocol. (Other protocols are not vulnerable because this area had been strengthened in TLS 1.0.) Conceptually, the vulnerability is very similar to the 2011 BEAST exploit. In order to successfully exploit POODLE the attacker must be able to inject malicious JavaScript into the victim's browser and also be able to observe and manipulate encrypted network traffic on the wire. As far as MITM attacks go, this one is complicated, but easier to execute than BEAST because it doesn't require any special browser plugins. If you care to learn the details, you can find them in the short paper or in Adam Langley's blog post.

SSL Labs Changes

We made three improvements to the SSL Labs web site to properly test and warn about the POODLE attack: 1) warnings about SSL 3 support and vulnerability to POODLE, 2) test for TLS_FALLBACK_SCSV and 3) new client test that detects support for SSL 3. At this time, a server vulnerable to the POODLE attack will be given a C grade, but we're likely to change this grading in the near future, after we carefully consider all our options.

What Now?

POODLE is a protocol-level vulnerability that can't be easily fixed. Although it might be possible to attempt a BEAST-style



ALPACA: ataque de confusión de protocolo sobre TLS

10 jun. 2021, 12:58:00 2 comentarios
internet, vulnerabilidades

Un grupo de académicos de la Universidad de Ruhr en Bochum, la Universidad de Ciencias Aplicadas de Münster y la Universidad de Paderborn han revelado un nuevo tipo de ataque, que aprovecha las configuraciones incorrectas de seguridad de la capa de transporte (TLS) en los servidores, para redirigir el tráfico HTTPS desde el navegador de la víctima a un punto final diferente, en otra dirección IP, y permite robar información confidencial.

Los ataques han sido denominados ALPACA, abreviatura de *"Application Layer Protocol Confusion - Analyzing and mitigating Cracks in TLS Authentication"*

"Los atacantes pueden redirigir el tráfico de un subdominio a otro, lo que resulta en una sesión TLS válida", dice el estudio. *"Esto rompe la autenticación de TLS y los ataques entre protocolos pueden ser posibles cuando el comportamiento de un servicio de protocolo puede comprometer al otro en la capa de aplicación"*.

TLS es un protocolo criptográfico que sustenta varios protocolos de capa de aplicación como HTTPS, SMTP, IMAP, POP3 y FTP para asegurar las comunicaciones a través de una red con el objetivo de agregar una capa de autenticación y preservar la integridad de los datos intercambiados mientras están en tránsito.

Los ataques ALPACA son posibles porque TLS no vincula una conexión TCP al protocolo de capa de aplicación previsto, explicaron los investigadores. Por lo tanto, se podría abusar de la falla de TLS para proteger la integridad de la conexión TCP para *"redirigir el tráfico TLS para el punto final y protocolo del servicio TLS previsto a otro punto final sustituto"*.

```
POST /
Host: www.bank.com
Cookie: secret
HELP <script>reflect()</script>
```

```
HTTP/1.1 200 OK
<script>stored()</script>
```

```
Unknown command:
<script>reflect();</script>
```

```
Cookie: secret
```

https://blog.segu-info.com.ar/2021/06/alpaca-ataque-de-confusion-de-protocolo.html

Seguridad en IP: TLS

HSTS

SSL Inspection

Certificate Pinning

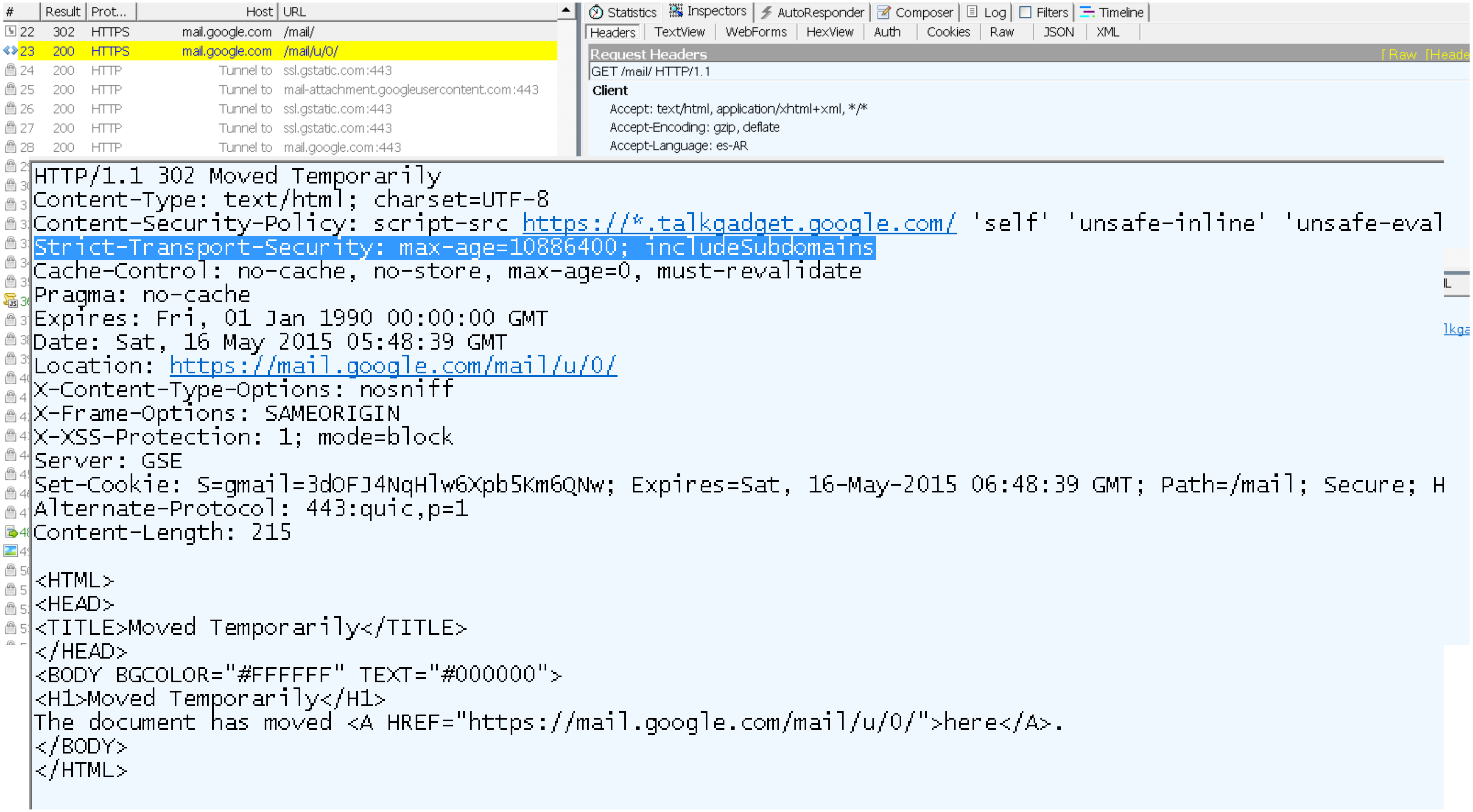
CAA, DANE, **OCSP Stapling**

HSTS **HTTP Strict Transport Security**

- El servidor informa que la única forma de interactuar es por SSL.
- Campo de la cabecera Strict-Transport-Security

- Después de su activación, el navegador no permite ninguna comunicación insegura con el sitio web que la utilizo previamente.

HSTS en la cabecera



#	Result	Prot...	Host	URL
22	302	HTTPS	mail.google.com	/mail/
23	200	HTTPS	mail.google.com	/mail/u/0/
24	200	HTTP	Tunnel to	ssl.gstatic.com:443
25	200	HTTP	Tunnel to	mail-attachment.googleusercontent.com:443
26	200	HTTP	Tunnel to	ssl.gstatic.com:443
27	200	HTTP	Tunnel to	ssl.gstatic.com:443
28	200	HTTP	Tunnel to	mail.google.com:443

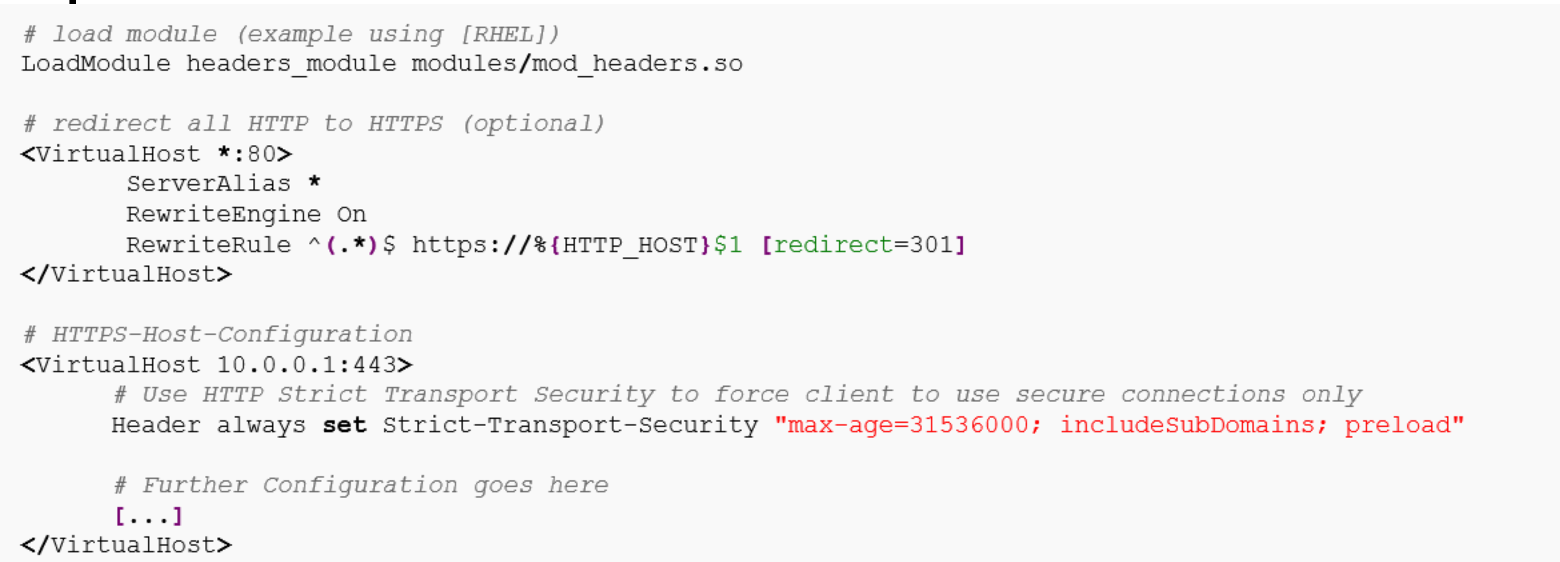
Request Headers

```
GET /mail/ HTTP/1.1
Client
    Accept: text/html, application/xhtml+xml, */*
    Accept-Encoding: gzip, deflate
    Accept-Language: es-AR
```

```
HTTP/1.1 302 Moved Temporarily
Content-Type: text/html; charset=UTF-8
Content-Security-Policy: script-src https://*.talkgadget.google.com/ 'self' 'unsafe-inline' 'unsafe-eval
Strict-Transport-Security: max-age=10886400; includeSubdomains
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: Fri, 01 Jan 1990 00:00:00 GMT
Date: Sat, 16 May 2015 05:48:39 GMT
Location: https://mail.google.com/mail/u/0/
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Server: GSE
Set-Cookie: S=gmail=3dOFJ4NqHlw6Xpb5Km6QNw; Expires=Sat, 16-May-2015 06:48:39 GMT; Path=/mail; Secure; H
Alternate-Protocol: 443:quic,p=1
Content-Length: 215

<HTML>
<HEAD>
<TITLE>Moved Temporarily</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<H1>Moved Temporarily</H1>
The document has moved <A HREF="https://mail.google.com/mail/u/0/">here</A>.
</BODY>
</HTML>
```

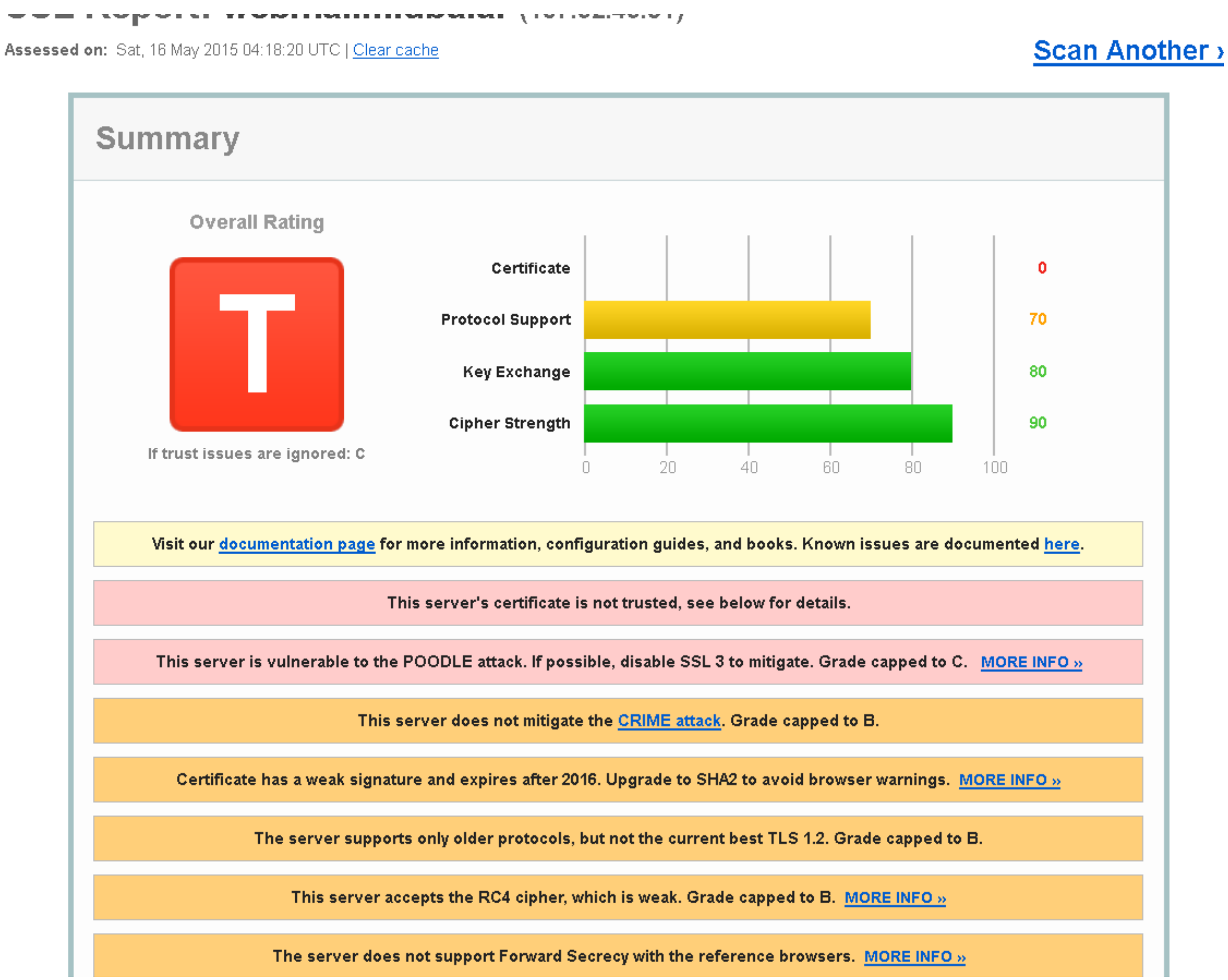
HSTS Configuración de apache

Apache



```
# load module (example using [RHEL])
LoadModule headers_module modules/mod_headers.so

# redirect all HTTP to HTTPS (optional)
<VirtualHost *:80>
       ServerAlias *
       RewriteEngine On
       RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
</VirtualHost>

# HTTPS-Host-Configuration
<VirtualHost 10.0.0.1:443>
      # Use HTTP Strict Transport Security to force client to use secure connections only
      Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"

      # Further Configuration goes here
      [...]
</VirtualHost>
```

Validación con Herramientas



Assessed on: Sat, 16 May 2015 04:18:20 UTC | Clear cache

Scan Another ›

Summary

Overall Rating

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server's certificate is not trusted, see below for details.

This server is vulnerable to the POODLE attack. If possible, disable SSL 3 to mitigate. Grade capped to C. MORE INFO »

This server does not mitigate the CRIME attack. Grade capped to B.

Certificate has a weak signature and expires after 2016. Upgrade to SHA2 to avoid browser warnings. MORE INFO »

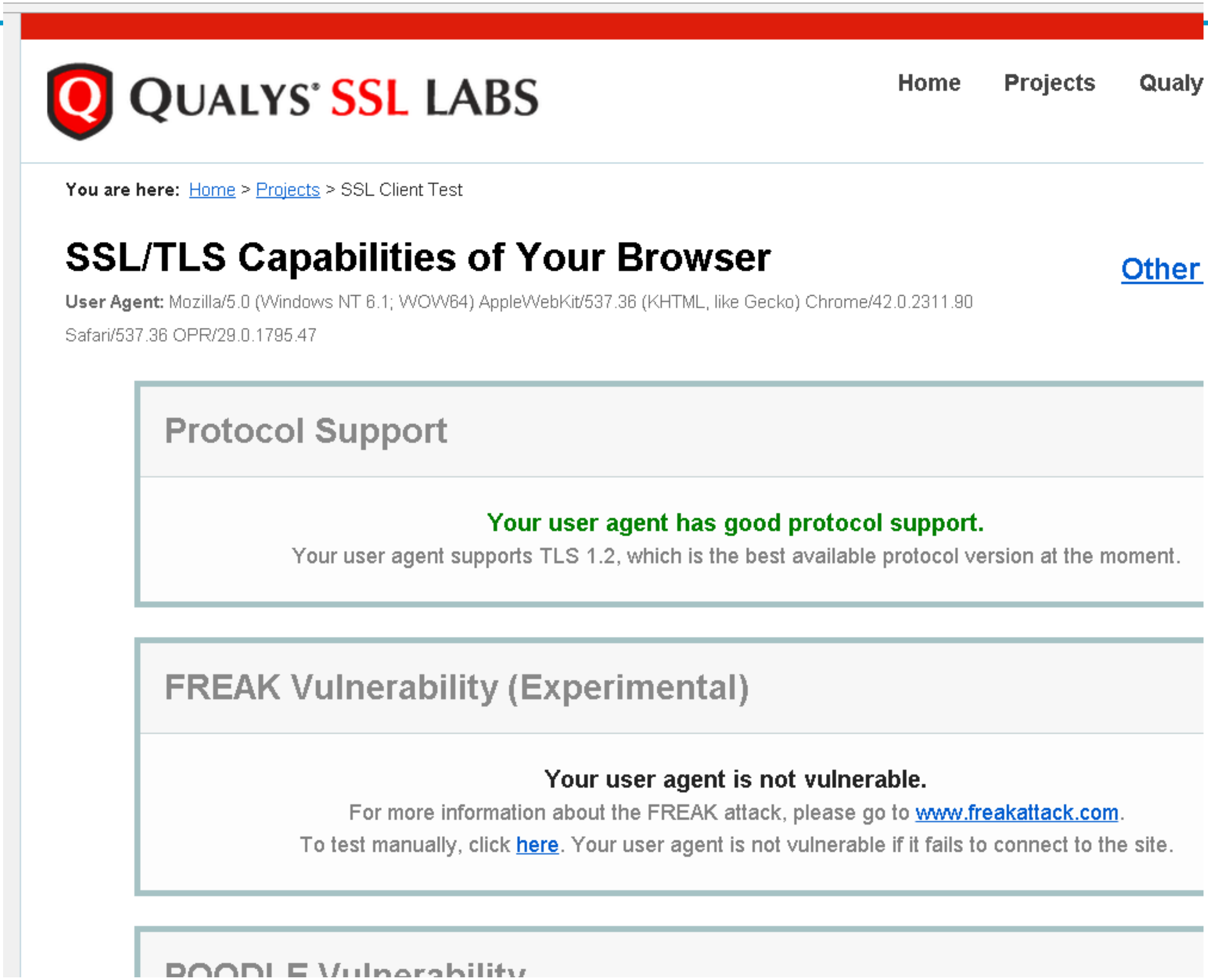
The server supports only older protocols, but not the current best TLS 1.2. Grade capped to B.

This server accepts the RC4 cipher, which is weak. Grade capped to B. MORE INFO »

The server does not support Forward Secrecy with the reference browsers. MORE INFO »

Fuente: https://www.ssllabs.com/ssltest/analyze.html?d=servidor.com.ar

Validación con Herramientas



You are here: Home > Projects > SSL Client Test

SSL/TLS Capabilities of Your Browser

Other

User Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.90 Safari/537.36 OPR/29.0.1795.47

Protocol Support

Your user agent has good protocol support.

Your user agent supports TLS 1.2, which is the best available protocol version at the moment.

FREAK Vulnerability (Experimental)

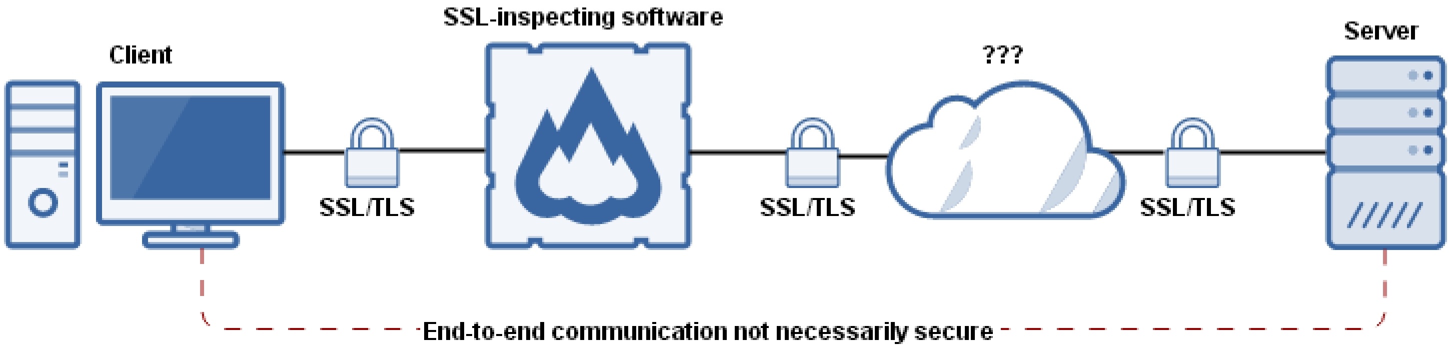
Your user agent is not vulnerable.

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

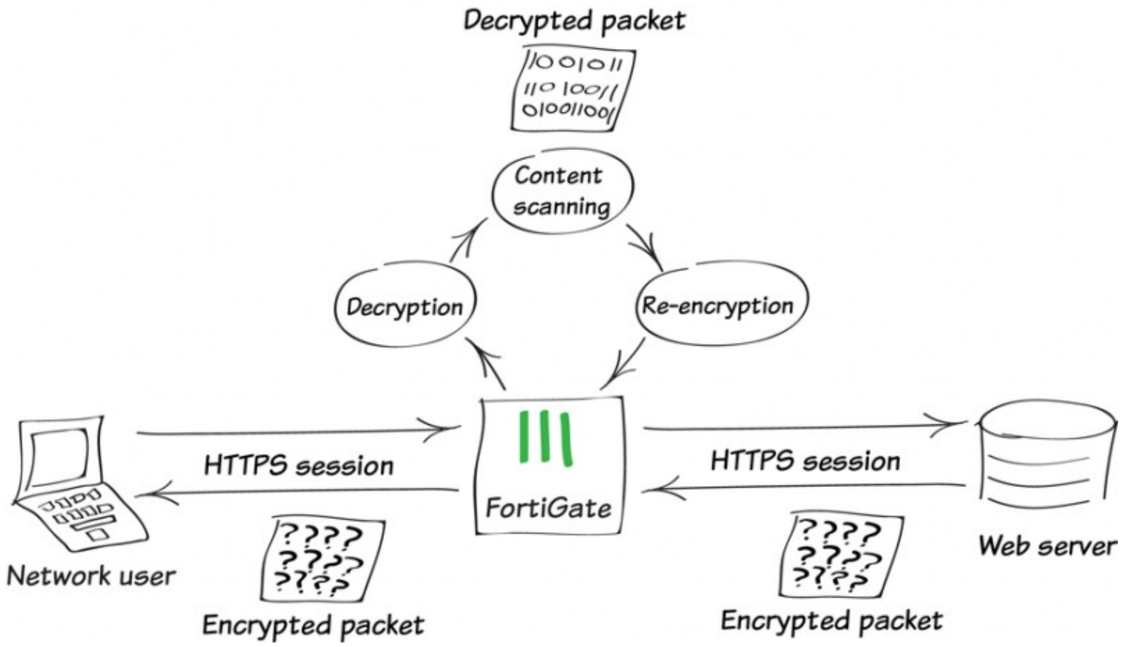
SSL Inspection



* Detectar Malware

- DLP
- Control de acceso a paginas basado en contenido

SSL Inspection

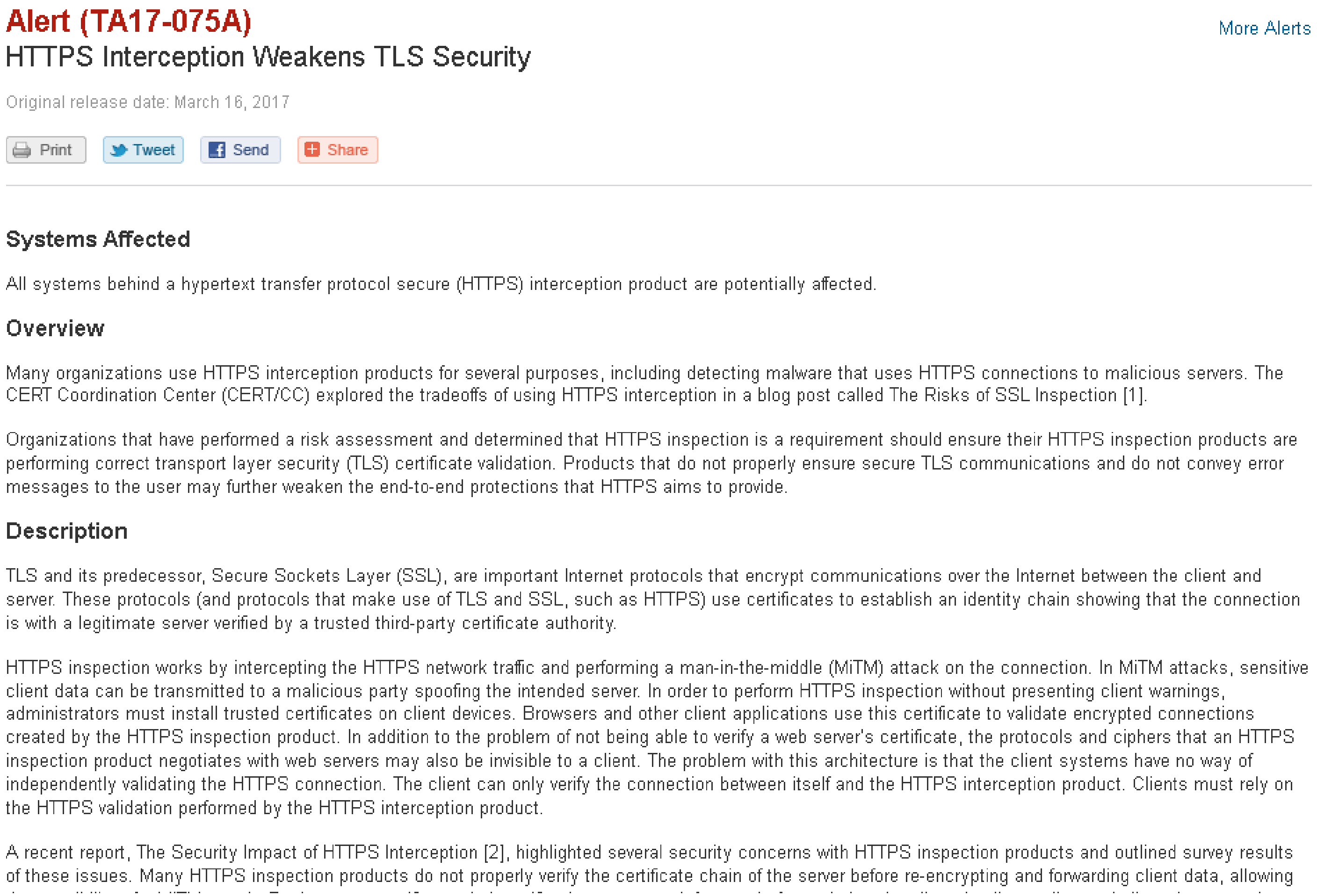


El equipo se hace pasar por el destino.

El cliente debe confiar en una CA que tiene el proxy o cargar certificados de los servicios que se inspeccionan. (para no tener warnings)

http://cookbook.fortinet.com/why-you-should-use-ssl-inspection/

SSL Inspection **Alert (TA17-075A)**



Alert (TA17-075A)

HTTPS Interception Weakens TLS Security

Original release date: March 16, 2017

Systems Affected

All systems behind a hypertext transfer protocol secure (HTTPS) interception product are potentially affected.

Overview

Many organizations use HTTPS interception products for several purposes, including detecting malware that uses HTTPS connections to malicious servers. The CERT Coordination Center (CERT/CC) explored the tradeoffs of using HTTPS interception in a blog post called The Risks of SSL Inspection [1].

Organizations that have performed a risk assessment and determined that HTTPS inspection is a requirement should ensure their HTTPS inspection products are performing correct transport layer security (TLS) certificate validation. Products that do not properly ensure secure TLS communications and do not convey error messages to the user may further weaken the end-to-end protections that HTTPS aims to provide.

Description

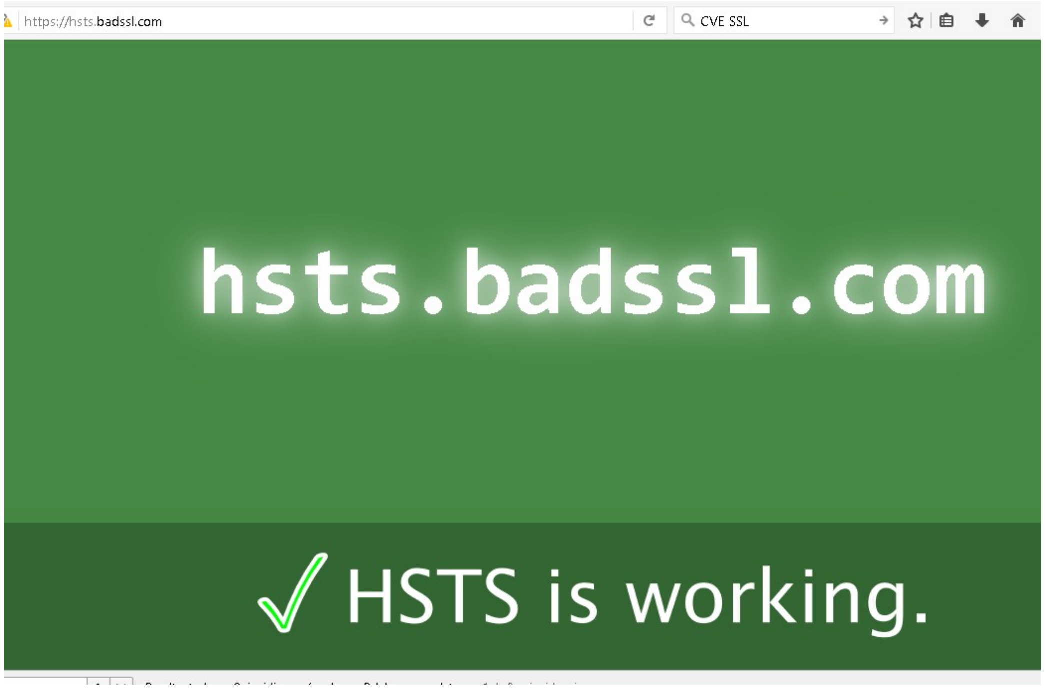
TLS and its predecessor, Secure Sockets Layer (SSL), are important Internet protocols that encrypt communications over the Internet between the client and server. These protocols (and protocols that make use of TLS and SSL, such as HTTPS) use certificates to establish an identity chain showing that the connection is with a legitimate server verified by a trusted third-party certificate authority.

HTTPS inspection works by intercepting the HTTPS network traffic and performing a man-in-the-middle (MiTM) attack on the connection. In MiTM attacks, sensitive client data can be transmitted to a malicious party spoofing the intended server. In order to perform HTTPS inspection without presenting client warnings, administrators must install trusted certificates on client devices. Browsers and other client applications use this certificate to validate encrypted connections created by the HTTPS inspection product. In addition to the problem of not being able to verify a web server's certificate, the protocols and ciphers that an HTTPS inspection product negotiates with web servers may also be invisible to a client. The problem with this architecture is that the client systems have no way of independently validating the HTTPS connection. The client can only verify the connection between itself and the HTTPS interception product. Clients must rely on the HTTPS validation performed by the HTTPS interception product.

A recent report, The Security Impact of HTTPS Interception [2], highlighted several security concerns with HTTPS inspection products and outlined survey results of these issues. Many HTTPS inspection products do not properly verify the certificate chain of the server before re-encrypting and forwarding client data, allowing

https://www.us-cert.gov/ncas/alerts/TA17-075A

SSL Inspection



hsts.badssl.com

✓ HSTS is working.

https://insights.sei.cmu.edu/cert/2015/03/the-risks-of-ssl-inspection.html
https://badssl.com/

Certificate Pinning

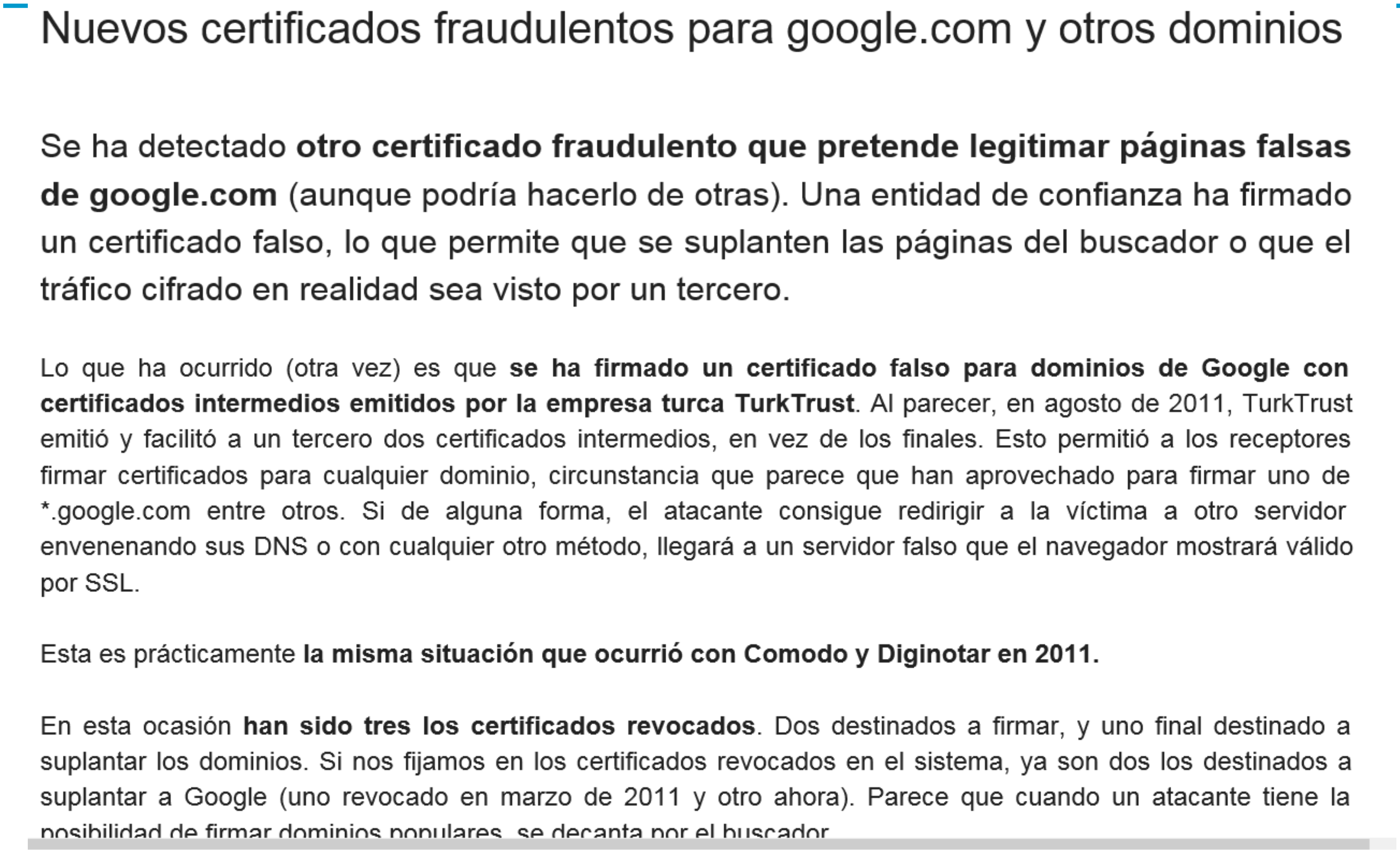
Técnica que permite detectar posibles certificados maliciosos, pero que respetan la cadena de confianza.

- **Para http, HPKP (HTTP Public Key Pinning)**

Referencias:

https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
http://blog.elevenpaths.com/2013/08/certificate-pinning-el-que-el-como-y-el.html
http://www.rfc-editor.org/rfc/rfc7469.txt

Certificate Pinning: ¿Por qué?



Nuevos certificados fraudulentos para google.com y otros dominios

Se ha detectado **otro certificado fraudulento que pretende legitimar páginas falsas de google.com** (aunque podría hacerlo de otras). Una entidad de confianza ha firmado un certificado falso, lo que permite que se suplanten las páginas del buscador o que el tráfico cifrado en realidad sea visto por un tercero.

Lo que ha ocurrido (otra vez) es que **se ha firmado un certificado falso para dominios de Google con certificados intermedios emitidos por la empresa turca TurkTrust**. Al parecer, en agosto de 2011, TurkTrust emitió y facilitó a un tercero dos certificados intermedios, en vez de los finales. Esto permitió a los receptores firmar certificados para cualquier dominio, circunstancia que parece que han aprovechado para firmar uno de *.google.com entre otros. Si de alguna forma, el atacante consigue redirigir a la víctima a otro servidor envenenando sus DNS o con cualquier otro método, llegará a un servidor falso que el navegador mostrará válido por SSL.

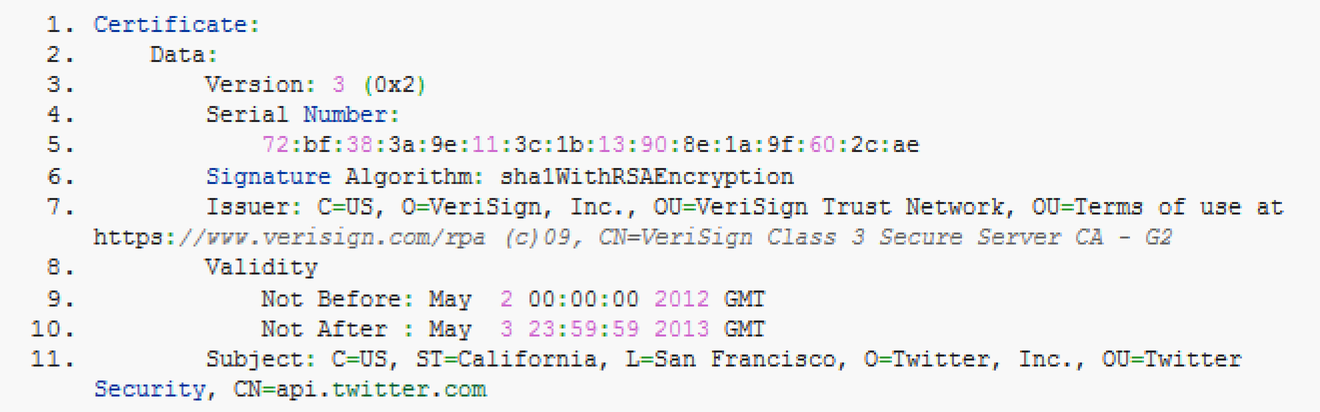
Esta es prácticamente **la misma situación que ocurrió con Comodo y Diginotar en 2011.**

En esta ocasión **han sido tres los certificados revocados**. Dos destinados a firmar, y uno final destinado a suplantar los dominios. Si nos fijamos en los certificados revocados en el sistema, ya son dos los destinados a suplantar a Google (uno revocado en marzo de 2011 y otro ahora). Parece que cuando un atacante tiene la posibilidad de firmar dominios populares, se decanta por el buscador.

http://unaaldia.hispasec.com/2013/01/nuevos-certificados-fraudulentos-para.html

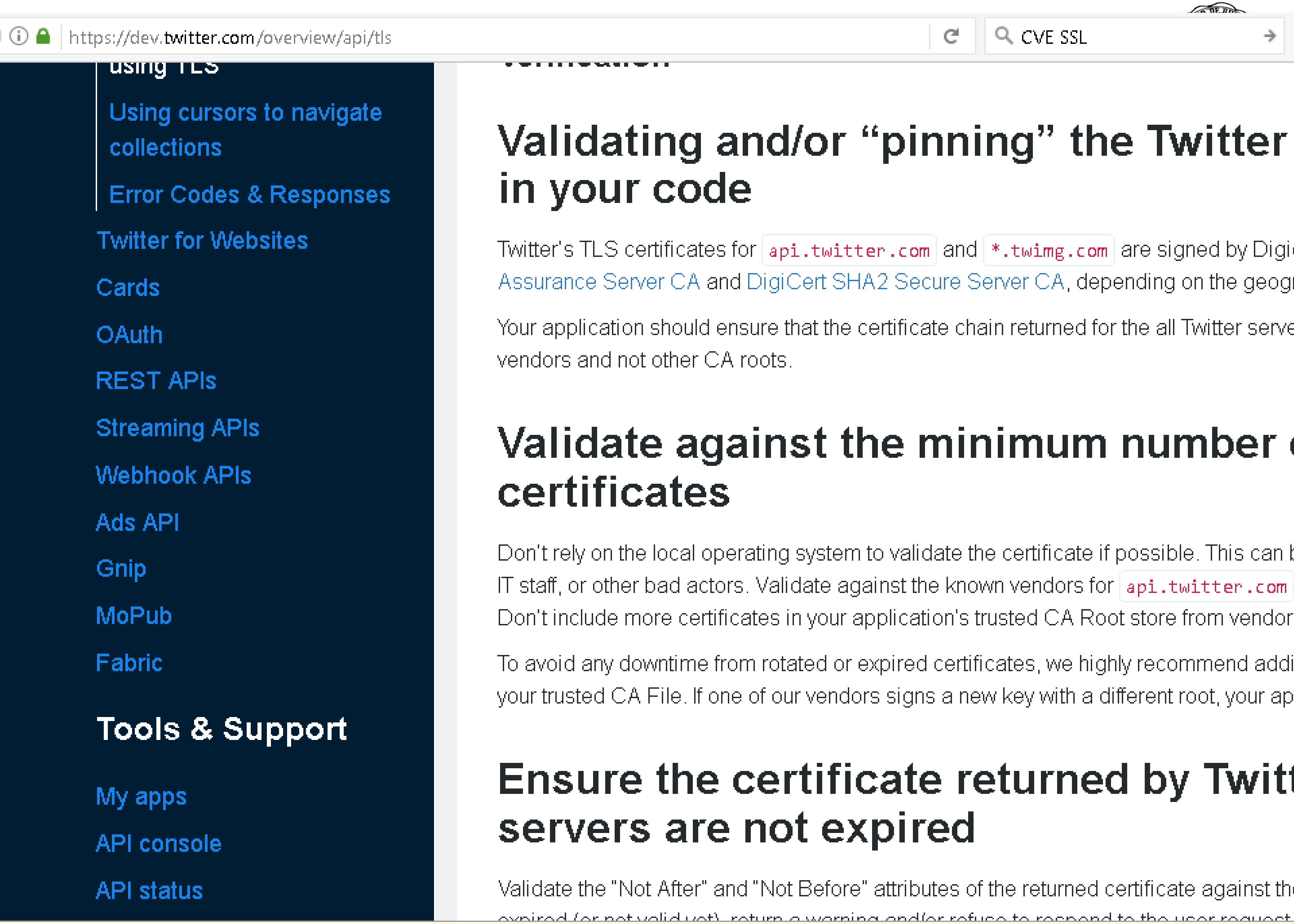
Certificate Pinning

Tiene como objetivo que una aplicación pueda asegurarse de que la cadena de certificados provista por todos los servidores de por ejemplo Twitter está firmada por uno de sus proveedores de certificados aprobados (Verisign EV a twitter.com, Verisign para api.twitter.com y Digicert para los demás) y no otras CA.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            72:bf:38:3a:9e:11:3c:1b:13:90:8e:1a:9f:60:2c:ae
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, OU=Terms of use at
https://www.verisign.com/rpa (c)09, CN=VeriSign Class 3 Secure Server CA - G2
        Validity
            Not Before: May  2 00:00:00 2012 GMT
            Not After : May  3 23:59:59 2013 GMT
        Subject: C=US, ST=California, L=San Francisco, O=Twitter, Inc., OU=Twitter
Security, CN=api.twitter.com
```

Fuente: https://dev.twitter.com/docs/security/using-ssl



using TLS

Using cursors to navigate collections

Error Codes & Responses

Twitter for Websites

Cards

OAuth

REST APIs

Streaming APIs

Webhook APIs

Ads API

Gnip

MoPub

Fabric

Tools & Support

My apps

API console

API status

Validating and/or “pinning” the Twitter in your code

Twitter's TLS certificates for `api.twitter.com` and `*.twimg.com` are signed by Digi Assurance Server CA and DigiCert SHA2 Secure Server CA, depending on the geogr

Your application should ensure that the certificate chain returned for the all Twitter serve vendors and not other CA roots.

Validate against the minimum number certificates

Don't rely on the local operating system to validate the certificate if possible. This can b IT staff, or other bad actors. Validate against the known vendors for `api.twitter.com` Don't include more certificates in your application's trusted CA Root store from vendor

To avoid any downtime from rotated or expired certificates, we highly recommend addi your trusted CA File. If one of our vendors signs a new key with a different root, your ap

Ensure the certificate returned by Twitt servers are not expired

Validate the "Not After" and "Not Before" attributes of the returned certificate against th expired (or not valid yet), return a warning and/or refuse to respond to the user request

Certificate Pinning: Public-Key-Pins RFC 7469

- Google lo incluyo en el chrome de forma estatica. De manera de impedir que alguien se impersone.
- El cliente conoce el hash del certificado. => Dificil y lento de mantener.
- Se propuso hpkp “*dynamic pinning*”
    - Cada sitio lo declara en la cabecera.
    - Se mantiene por un tiempo.
    - Mecanismo de renovación por CRL.

SPEC En
https://www.ietf.org/mail-archive/web/websec/current/pdfnSTRd9kYcY.pdf

Certificate Pinning

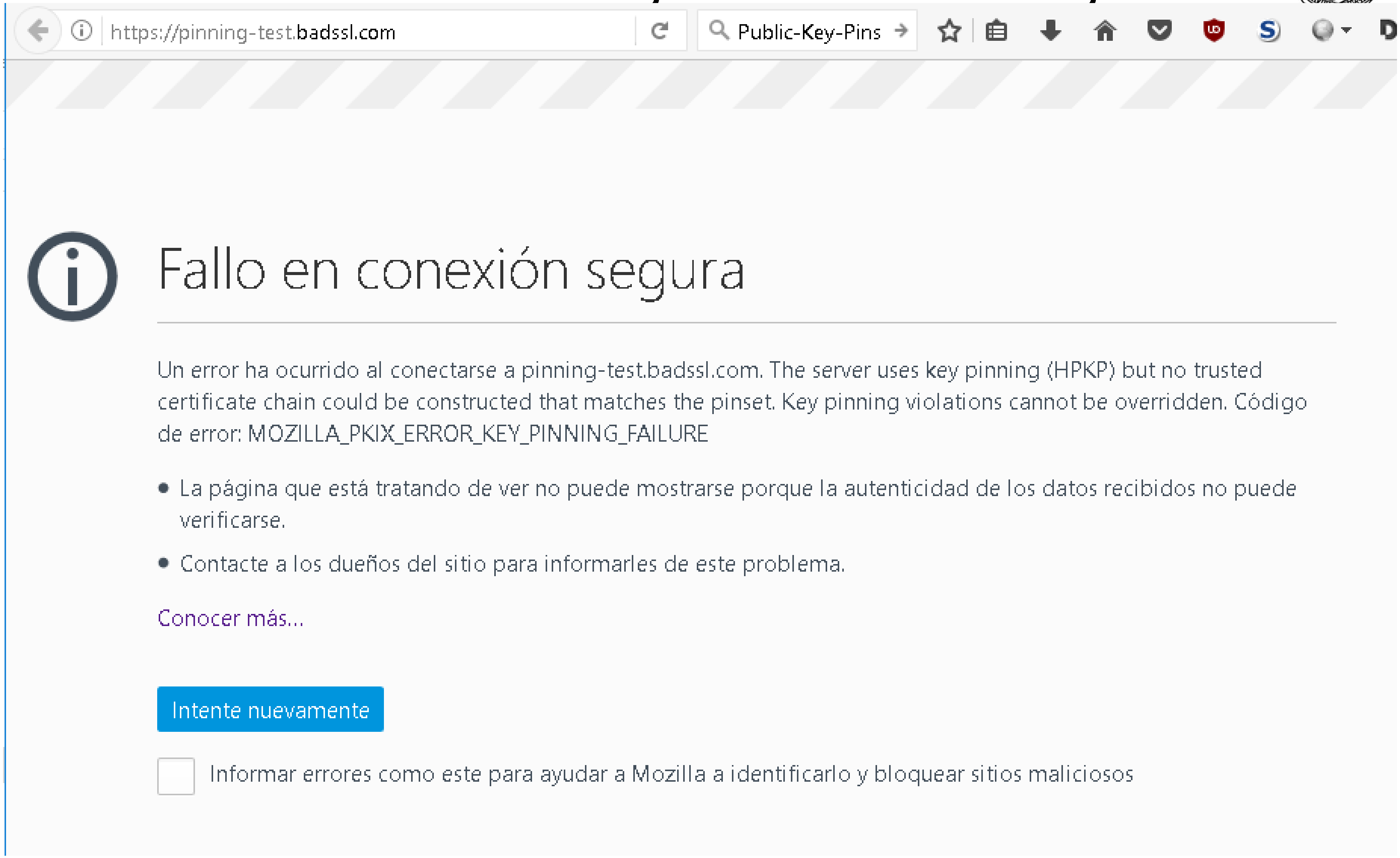
```
Public-Key-Pins:
        pin-sha256="hash cert=";
        pin-sha256=“hash de un cert BKP=";
        Max-age=5184000; (DOS MESES)
        includeSubDomains; (opc)
        report-uri=https://www.example.org/hpkp-report (opc)
```

Configuración en apache: Modulo mod_headers

```
Header always set Public-Key-Pins "pin-sha256=\"hashcert=\"; pin-sha256=\"base64+backup==\"; max-age=5184000; includeSubDomains“
```

https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning

Certificate Pinning: error del navegador



Fallo en conexión segura

Un error ha ocurrido al conectarse a pinning-test.badssl.com. The server uses key pinning (HPKP) but no trusted certificate chain could be constructed that matches the pinset. Key pinning violations cannot be overridden. Código de error: MOZILLA_PKIX_ERROR_KEY_PINNING_FAILURE

- La página que está tratando de ver no puede mostrarse porque la autenticidad de los datos recibidos no puede verificarse.
- Contacte a los dueños del sitio para informarles de este problema.

Conocer más…

□ Informar errores como este para ayudar a Mozilla a identificarlo y bloquear sitios maliciosos

Certificate Pinning: posible problema

- Se mantiene por un tiempo.
  - Esto hace por ejemplo que no pueda renovar ya que el hash cambia… se podría denegar servicio
  - Si no las perdes y solo las queres rotar. Igualmente es complicado porque los clientes tienen los hashes
- Alguien consigue acceso al sitio y pone el header y un certificado propio. Luego quita el header…
- En el caso del content inspection. Simplemente quito el header.

https://blog.qualys.com/ssllabs/2016/09/06/is-http-public-key-pinning-dead-

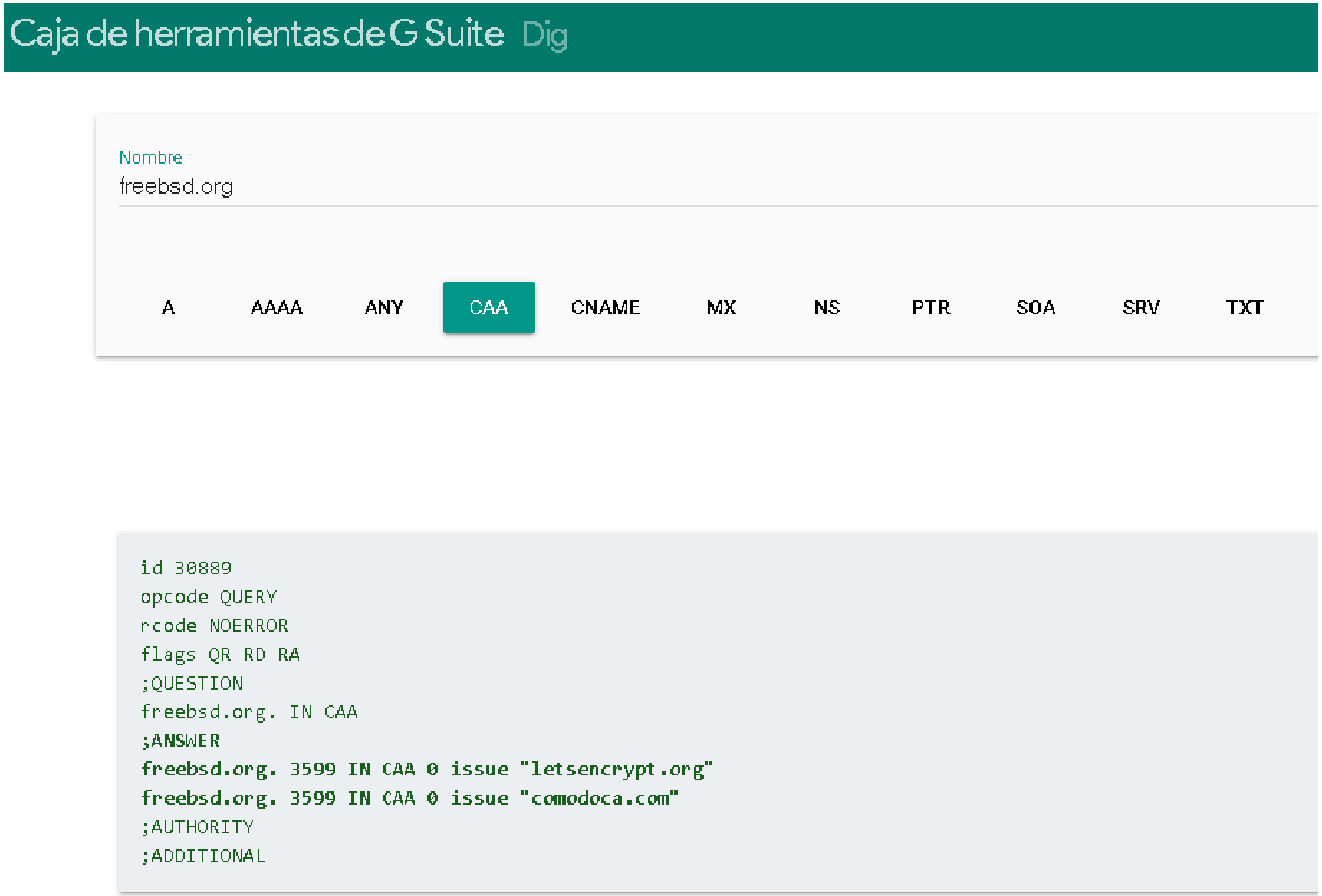
CAA DNS Certification Authority Authorization

DNS Resource Record

- Permite que cada dominio defina que CAs pueden emitirle certificados.
- Desde el 2017, las CAs están obligadas a consultar este valor antes de emitir un certificado.
- De no existir un registro CAA, cualquier entidad está autorizada a emitir certificados para ese dominio o subdominio.

https://tools.ietf.org/html/rfc6844

CAA DNS como verificarlo



Nombre
freebsd.org

```
id 30889
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
freebsd.org. IN CAA
;ANSWER
freebsd.org. 3599 IN CAA 0 issue "letsencrypt.org"
freebsd.org. 3599 IN CAA 0 issue "comodoca.com"
;AUTHORITY
;ADDITIONAL
```

https://toolbox.googleapps.com/apps/dig/#CAA/

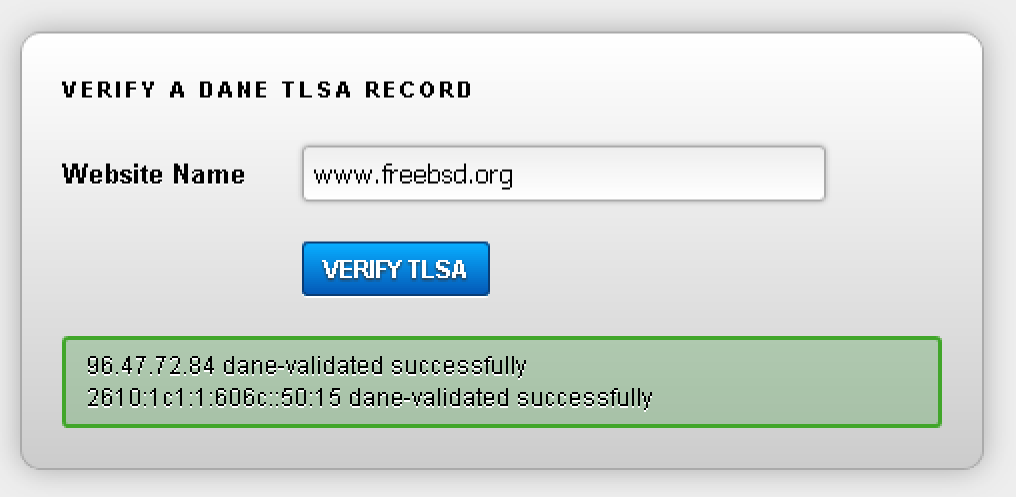
DANE DNS-Based Authentication of Named Entities

- **Registro DNS “**TLSA”
- **Securing the Association of a Domain Name with a Server's Certificate**

https://www.internetsociety.org/blog/2013/12/want-to-quickly-create-a-tlsa-record-for-dane-dnssec/

https://tools.ietf.org/html/rfc6698

DANE verificación online

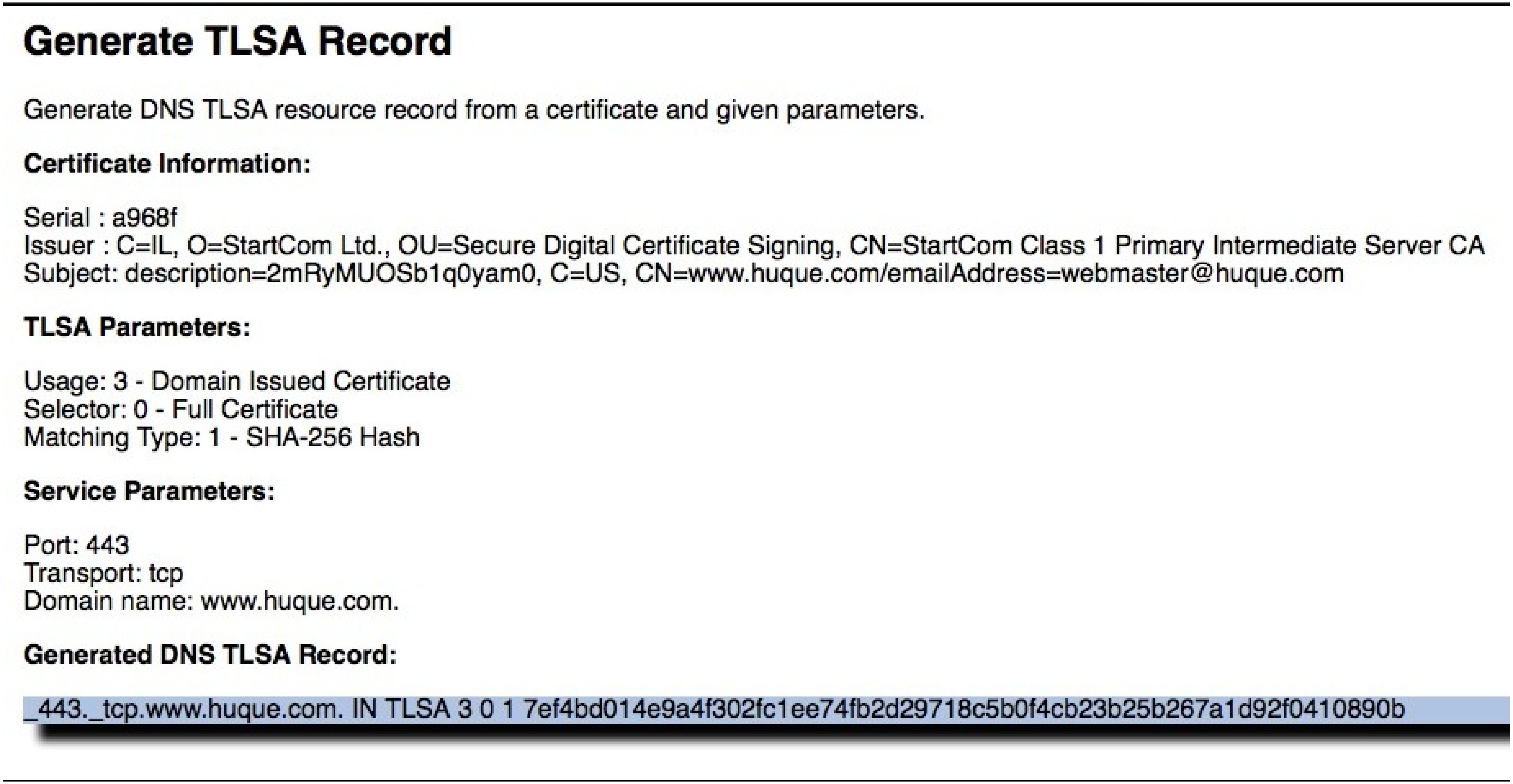


www.freebsd.org

96.47.72.84 dane-validated successfully
2610:1c1:1:606c::50:15 dane-validated successfully

https://check.sidnlabs.nl/dane/

DANE Ejemplo Registro TLSA



Generate TLSA Record

Generate DNS TLSA resource record from a certificate and given parameters.

Certificate Information:

Serial : a968f
Issuer : C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing, CN=StartCom Class 1 Primary Intermediate Server CA
Subject: description=2mRyMUOSb1q0yam0, C=US, CN=www.huque.com/emailAddress=webmaster@huque.com

TLSA Parameters:

Usage: 3 - Domain Issued Certificate
Selector: 0 - Full Certificate
Matching Type: 1 - SHA-256 Hash

Service Parameters:

Port: 443
Transport: tcp
Domain name: www.huque.com.

Generated DNS TLSA Record:

```
_443._tcp.www.huque.com. IN TLSA 3 0 1 7ef4bd014e9a4f302fc1ee74fb2d29718c5b0f4cb23b25b267a1d92f0410890b
```

OCSP Stapling

- Es formalmente conocido como la extensión de Solicitud de estado de certificado TLS,
- Es un estándar que permite verificar el estado de revocación de los certificados digitales X.509 en forma mas rápida y eficiente durante la carga de una página WEB.
- Permite al sitio de un certificado proporcionar respuestas OCSP con sello de tiempo firmado por la CA durante el handshake TLS inicial
- Elimina la necesidad de los clientes de contactara la CA.
- Objetivo mejorar tanto la seguridad como el rendimiento.

https://tools.ietf.org/html/rfc6961

https://www.digicert.com/kb/ssl-support/apache-enable-ocsp-stapling-on-server.htm

Seguridad en IP: TLS

Mejores Prácticas

SSL and TLS Deployment Best Practices

TLS es una tecnología engañosamente simple. Es fácil de implementar y simplemente funciona, excepto cuando falla.

el cifrado no suele ser fácil de implementar correctamente.

Para garantizar que TLS brinde la seguridad necesaria, los administradores y desarrolladores de sistemas deben realizar un esfuerzo adicional para configurar adecuadamente sus servidores y desarrollar sus aplicaciones.

Desde 2009: 6 versiones la ultima del 15/01/2020

https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices

INTERACTUANDO CON SSL/TLS: Mejores Prácticas

- ✓ **Clave privada y el certificado**
  - ✓ Use las claves privadas de 2048 bits
  - ✓ Proteger las claves privadas
  - ✓ Considerar la cobertura suficiente de nombres de dominio
    - ✓ www.empresa.com www.empresa.com.ar
  - ✓ Obtenga certificados de una CA fiable
  - ✓ Use solo algoritmosde certificación fuertes (sha1 -> sha256)
  - ✓ Usar DNS CAA (Certification Authority Authorization)

- ✓ **Configuración**
  - ✓ Utilice Cadenas de certificación completas
  - ✓ Utilice únicamente protocolos seguros
  - ✓ Utilice sólo suites de cifrado seguras
  - ✓ Control de selección de suites de cifrado
  - ✓ Desactivar la renegociación iniciada por el cliente
  - ✓ Mitigar problemas conocidos
    - ✓ Desactivar la renegociación insegura
    - ✓ Deshabilitar la compresión TLS

- ✓ **Rendimiento**
  - ✓ No utilice claves privadas demasiado grandes
  - ✓ Asegúrese de que la reanudación de sesión sea posible.
  - ✓ Permitir conexiones persistentes (HTTP)
  - ✓ OCSP Stapling

Version 1.6-draft (15 January 2020)

Fuente: ***https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices***

INTERACTUANDO CON SSL/TLS: Mejores Prácticas

- ✓ **Diseño de aplicaciones (HTTP)**
  - ✓ Cifrar el 100% de su tráfico del sitio web
  - ✓ Eliminar contenido mixto
  - ✓ Comprender y reconocer la confianza de terceros
  - ✓ Asegúrese de que las cookies están asegurados
  - ✓ Habilitar HTTP Strict Transport Security (HSTS)
  - ✓ Deshabilitar el almacenamiento en caché de contenido confidencial
  - ✓ Asegúrese de que no hay otras vulnerabilidades

- ✓ **Validación continúa con Herramientas**

- ✓ **Topicos Avanzados**
  - ✓ Public Key Pinning
  - ✓ DNSSEC and DANE

Fuente: *https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices*