MindWorks UK

ITP Technology Transfer Partners

TLS

SSL/TLS for system admins

Making progress with crypto-gibberish

Joe Doupnik
Prof (retired), Univ of Oxford
jrd@netlab1.net
jdoupnik@microfocus.com
MindworksUK and Micro Focus

This material is best read carefully when we plunge into local improvements

The presentation is available on https://netlab1.net

UNIVERSITY OF OXFORD

Version 2.2

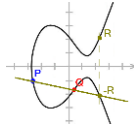
MICRO FOCUS

The subject is Internet communications security

Specifically
- encryption on the wire (crypto details, anti-snooping steps)
- verify the other end is what it proclaims (certs, DNS name)

Why? Because barbarians roam the countryside seeking victims and plunder.

Encryption involves complicated math & logic, with many choices of algorithms, and be opaque to outsiders. Certificates provide verifiable identity information and a PKI (Public Key Infrastructure) public & private key pair.



Sharing crypto keys is tricky; see Diffie-Hellman for cleverness. The SSL engine, frequently *openssl*, provides many crypto algorithms and we need to control it.

Rather than become crypto experts we rely upon good test facilities, read commentary about issues of interest, and configure our applications to employ strong **protocols** and **ciphers** with proper **controls**. Help with **certs** too. **pc^3**

This document is not a detailed manual. It is a guide with examples & pointers. Notice that “url” items are clickable for easy review of cited sources.

Coming into agreement about the task ahead

Creating a secure comms channel involves a series of detailed steps, a TLS protocol.

First, the client makes a TCP connection to the server with a structured **Hello** message in which are details such as its list of crypto algorithms, the DNS name of the desired server, and more. Optionally, ask for server's cert validity, offer a certificate to identify the client.

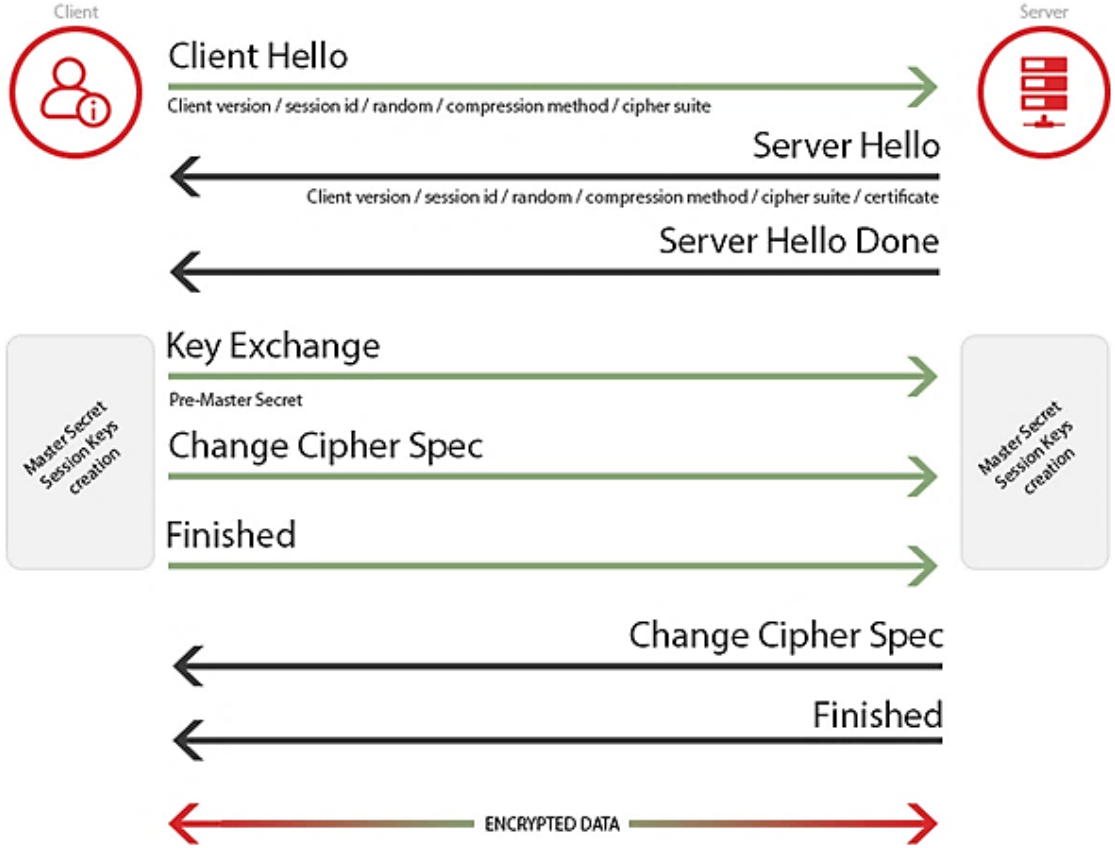
The server selects an algorithm which both sides can perform. That choice governs kind of *key exchange* and *encryption method*. Best is when the server chooses the order. In its **Hello reply** the server provides the crypto choice, its *certificate* with public key and identity, plus other important bits. This establishes The Rules.

Included is the ticklish matter of exchanging a crypto key(s) beyond the certificate PKI key pair while shielding from spying listeners and fakery by possible agents in the middle. This would make a good thriller book. 

Next are two illustrations about underlying complexity of starting a secure connection.

Then we examine test facilities and configuration of several applications. Ah, good.

TLS v1.2 handshake packet sequence, protocol social etiquette



Server cert is not protected, thus is vulnerable to change by relays

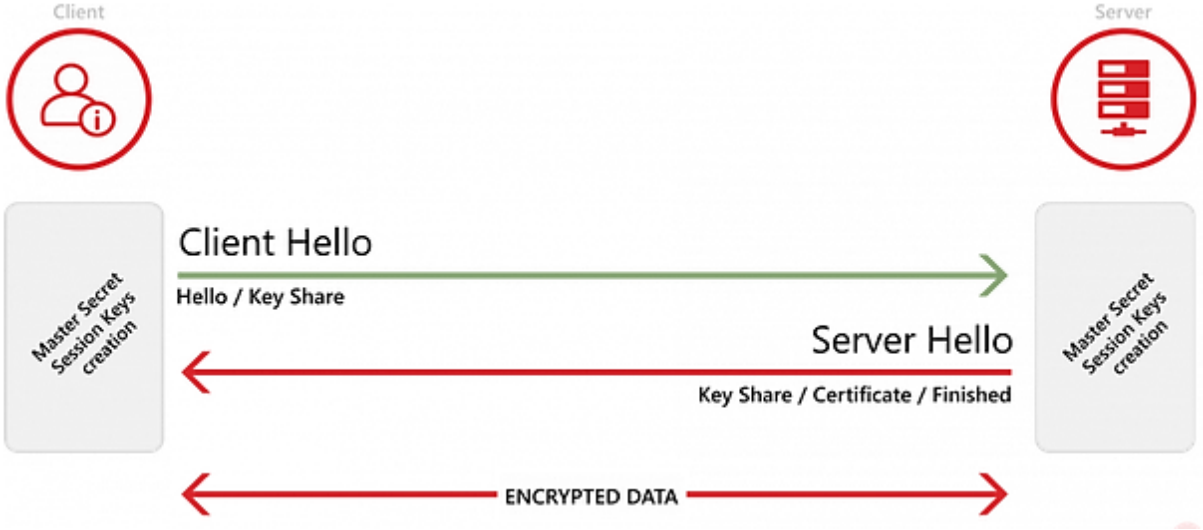
Knock knock
Do come in and sit down
Tea? Oh yes, thank you
Milk? Just a splash
Sugar? No, just plain
A nibble? Why yes please
And how is your sister?
Etc.

Chatty, many network hops

Finally, the real data

From https://www.acunetix.com/blog/articles/establishing-tls-ssl-connection-part-5/

TLS v1.3 handshake packet sequence, more business-like



Here is my Form 57

Adjusting it, stamp, approved. Copy for you.

Clearly this is much more compact than that of TLS 1.2 and prior. Rules are tightened to reduce negotiation chatter. The server's certificate is protected within the handshake. Put simply, be more efficient and protect better.

From https://www.acunetix.com/blog/articles/establishing-tls-ssl-connection-part-5/

Additional https://www.thesslstore.com/blog/tls-1-3-everything-possibly-needed-know/

That is all very interesting, but it does have rather many tricky details

After the handshake admin work the real encryption proceeds, as either a sequence of individually encrypted blocks (avoid CBC cipher block chaining) or as a stream.

Here there be dragons. No matter who creates an algorithm soon someone discovers an exposure method. Thus there is a long historical trail of methods, and alas we have to choose amongst them.

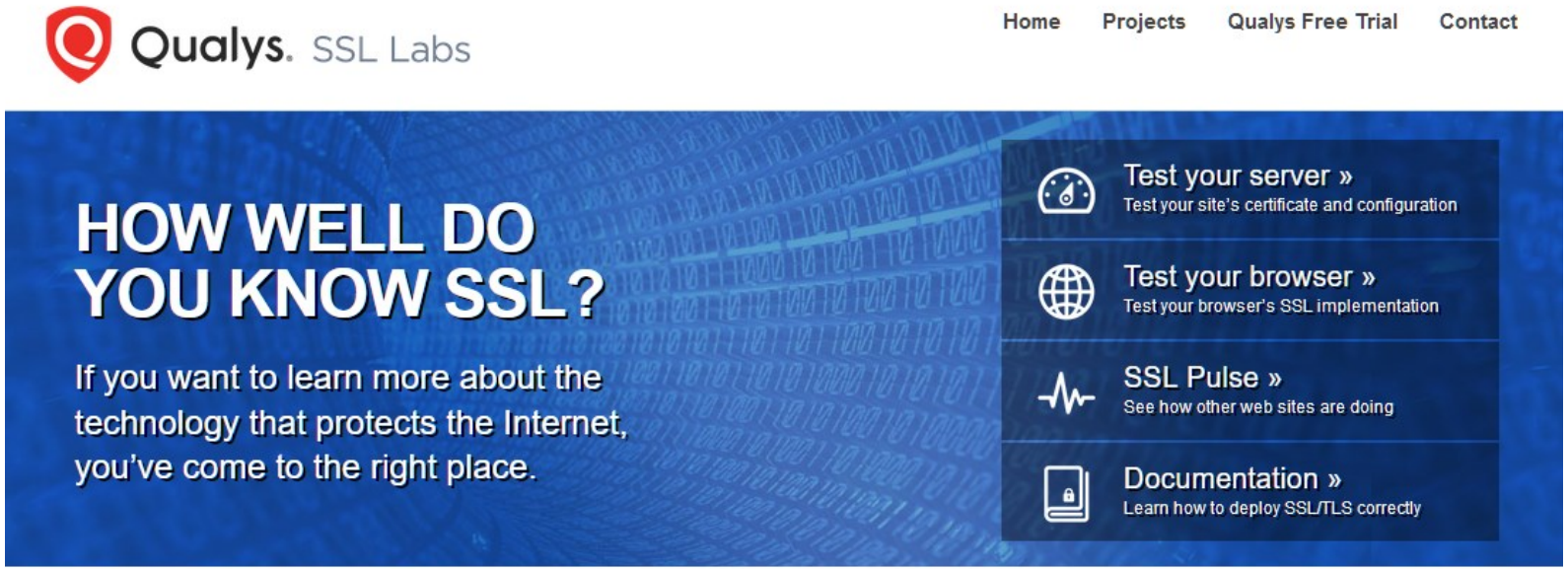
Our task, without becoming experts, is permit only the currently “good” methods. We employ testing tools & examples to indicate “good” then configure our applications. Fortunately, that is reasonable.

So then, let us take a guided tour together through

Nice dino, ah er crypto. Sit Stay

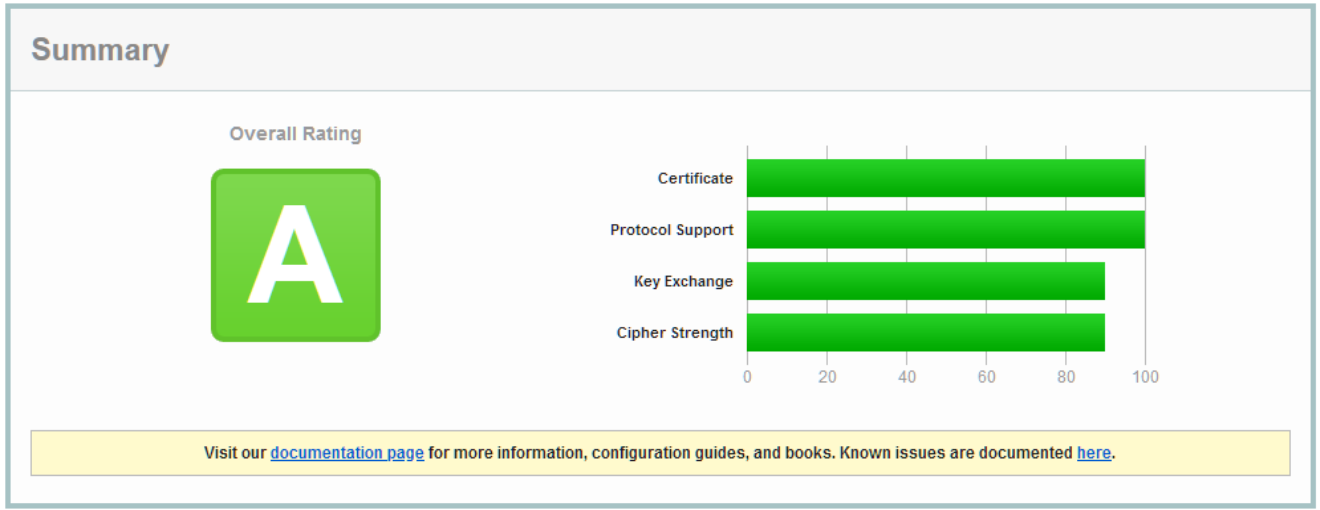
All aboard
Let’s begin

A primary test tool, Internet based free web testing



From https://www.ssllabs.com
Also review its document collection and implementation census

SSL Labs report about a well configured web site (TLS v1.2)

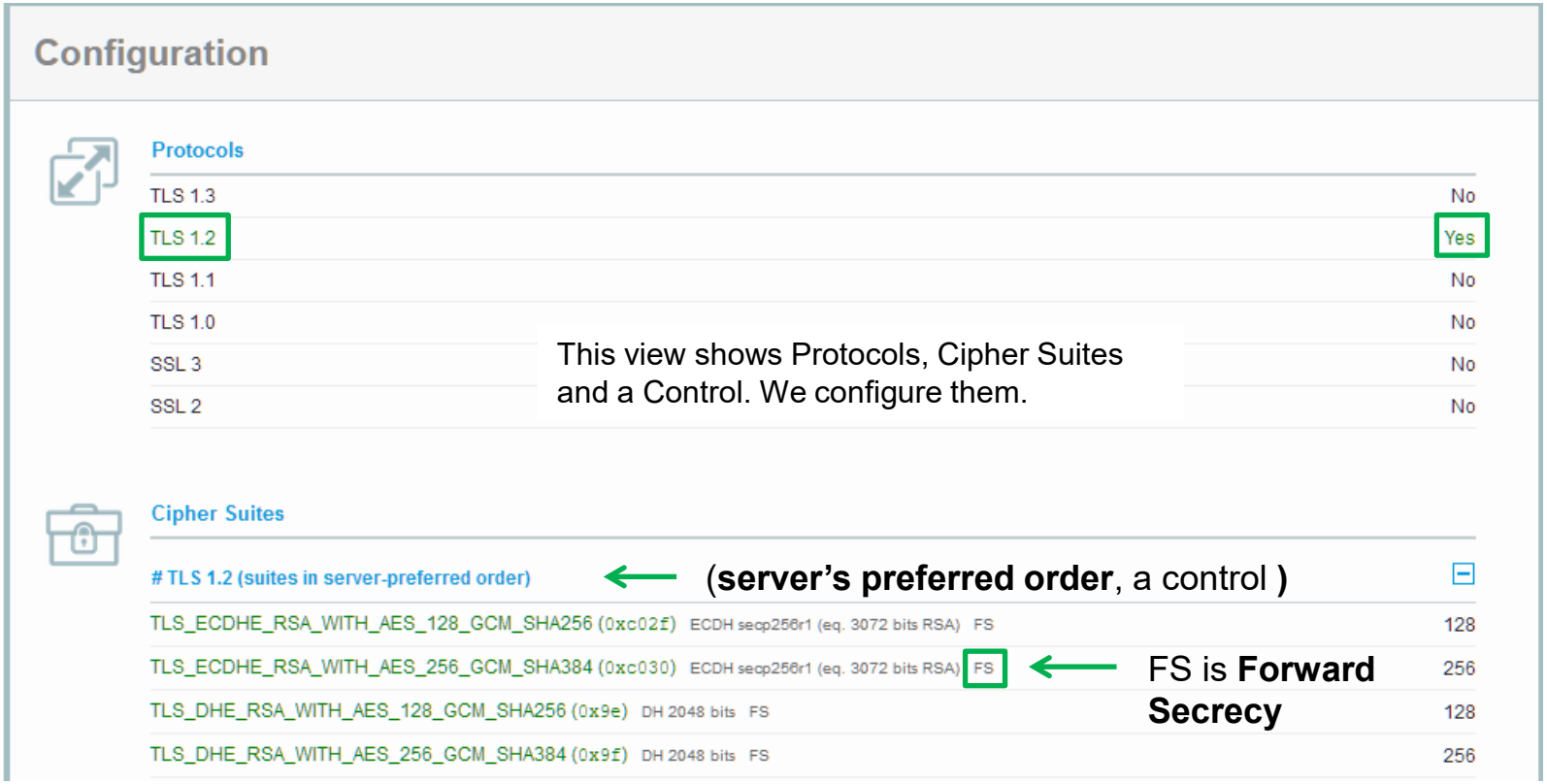


OES2018 SP2, configured

From https://www.ssllabs.com

Avoid adding HSTS for an ego-boosting-only A+ grade. HSTS is invasive to clients.

SSL Labs report about a well configured web site (TLS v1.2)



Green colour means advice of good/strong. We do notice this.

SSL Labs report about a well configured web site controls dept

Secure Renegotiation	Supported
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Mitigated server-side (more info)
POODLE (SSLv3)	No, SSL 3 not supported (more info)
POODLE (TLS)	No (more info)
Zombie POODLE	No (more info)
GOLDENDOODLE	No (more info)
OpenSSL 0-Length	No (more info)
Sleeping POODLE	No (more info)
Downgrade attack prevention	Unknown (requires support fo
SSL/TLS compression	No

Secure Renegotiation is good, but do not let clients initiate it

Compression by SSL/TLS, avoid

Please note carefully:
Controls improve security and robust comms. Usage is via each application which passes settings to the crypto engine.
Thus review application docs.

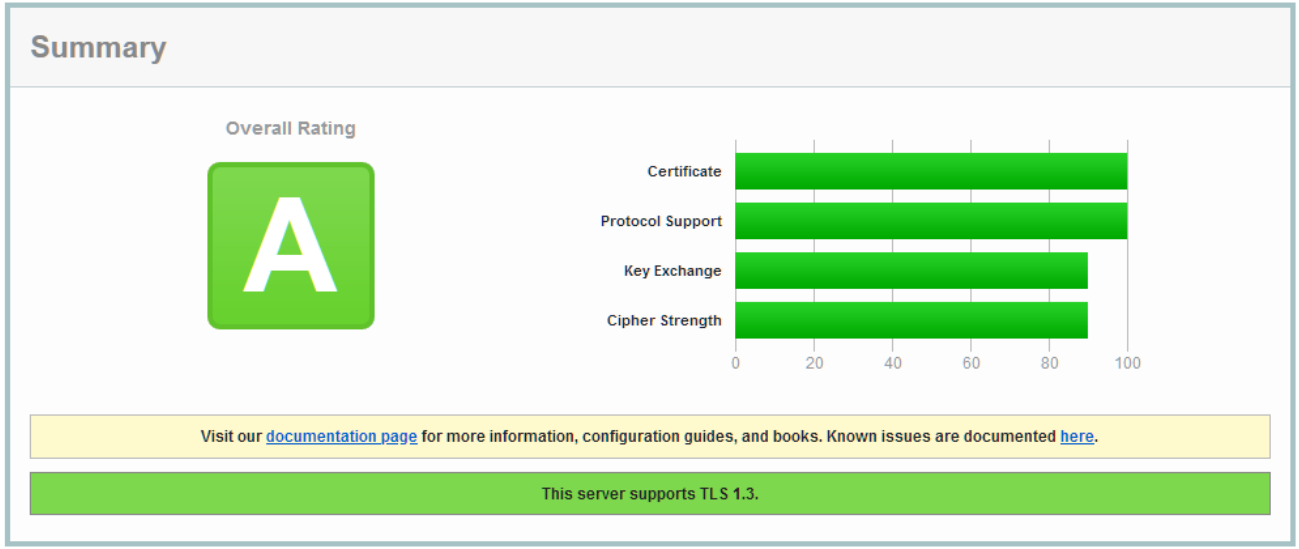
Forward Secrecy (change keys often) is very desirable, set by choice of crypto algorithm

Forward Secrecy	Yes (with most browsers) ROBUST (more info)
ALPN	Yes h2 http/1.1
NPN	No
Session resumption (caching)	Yes
Session resumption (tickets)	No
OCSP stapling	Yes
Strict Transport Security (HSTS)	No
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

Session resumption: tickets are less good than caching

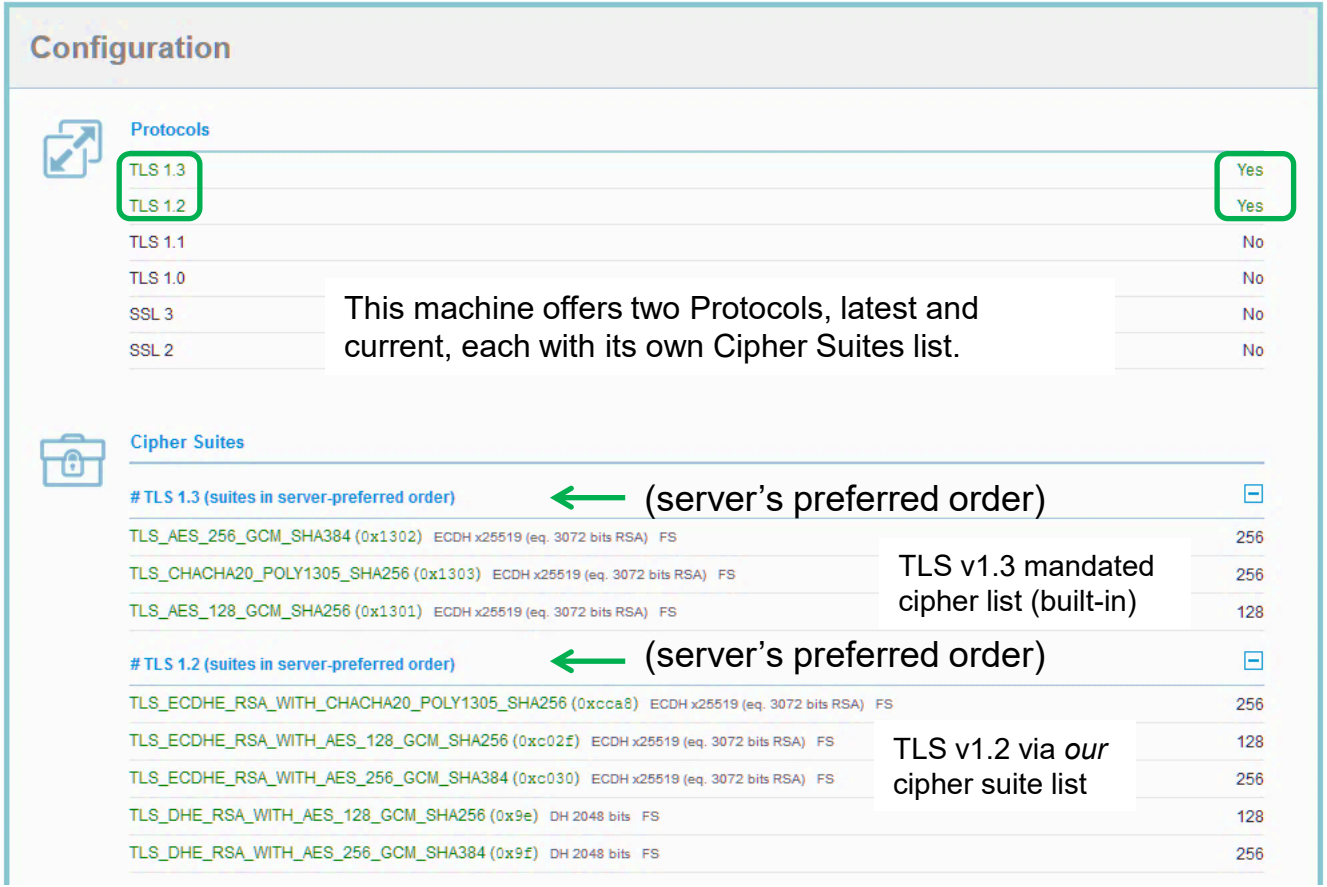
OCSP stapling is good
HSTS modifies clients, be wary

SSL Labs report about a modern web site (TLS v1.2, 1.3)



OpenSuSE LEAP 15.2, configured

SSL Labs report about a modern web (TLS v1.2, 1.3)

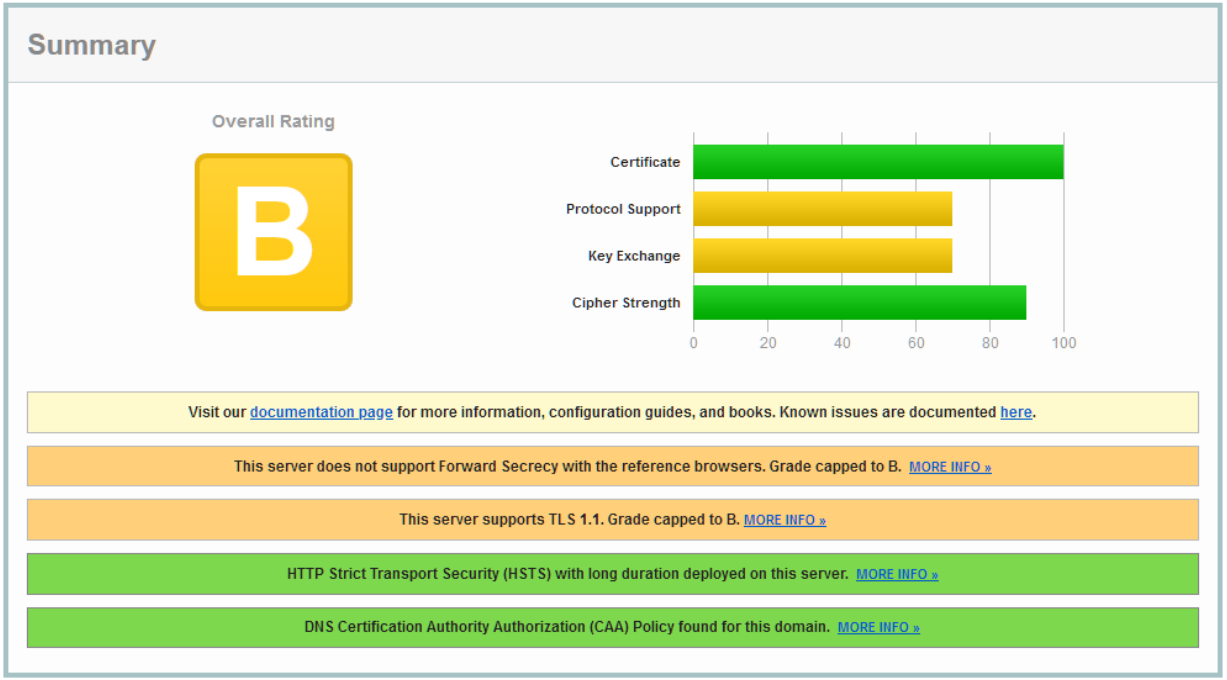


Latest TLS v1.3 and fallback v1.2

Two TLS version dependent cipher suites.

"FS" tag is Forward Secrecy, desirable

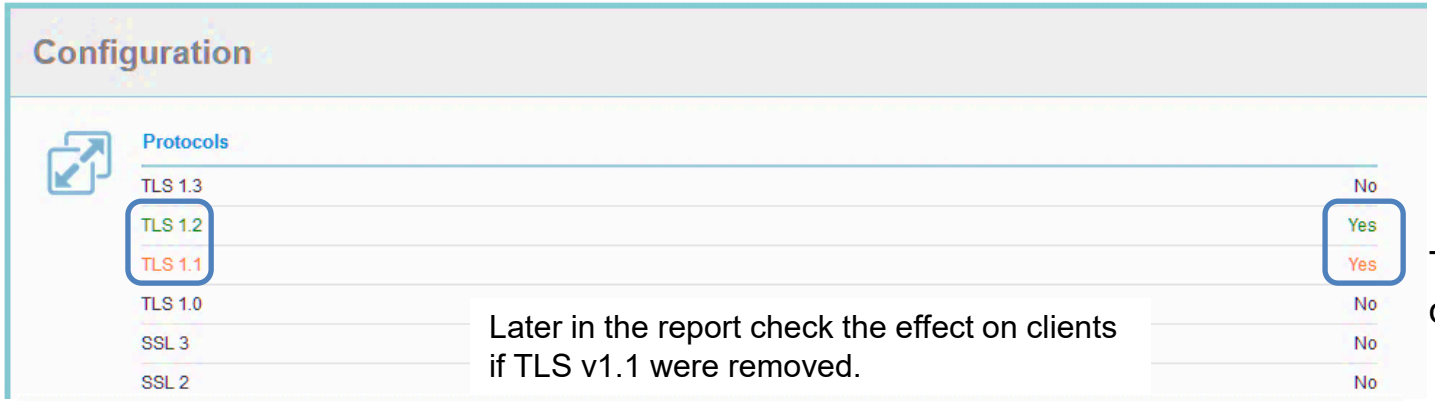
SSL Labs report about a typical site (TLS v1.1, 1.2)



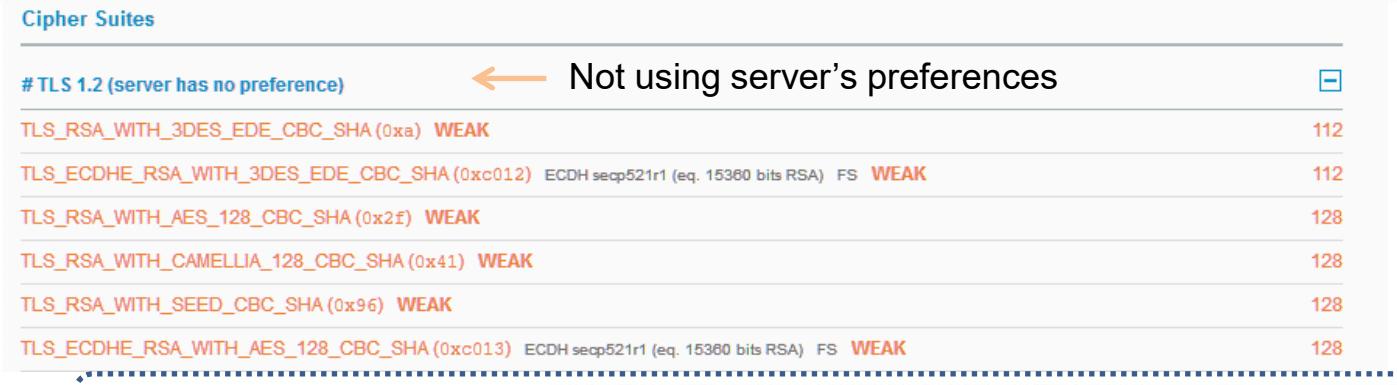
A typical test result

Warning comments

SSL Labs report about a typical site (TLS v1.1, 1.2)



TLS v1.1 is depreciated



The CBC suites are weak. Further down are stronger GCM, such as this choice

Example of old **CBC** vs new **GCM** cipher block chaining

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) ECDH secp521r1 (eq. 15360 bits RSA) FS WEAK

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) ECDH secp521r1 (eq. 15360 bits RSA) FS

Orange colour means advice of weakness. We take note.

SSL Labs report about a typical site (TLS v1.1, 1.2)

Secure Renegotiation	Supported
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Mitigated server-side (more info)
POODLE (SSLv3)	No, SSL 3 not supported (more info)
POODLE (TLS)	No (more info)
Zombie POODLE	No (more info) TLS 1.2 : 0x000a
GOLDENDOODLE	No (more info) TLS 1.2 : 0x000a
OpenSSL 0-Length	No (more info) TLS 1.2 : 0x000a
Sleeping POODLE	No (more info) TLS 1.2 : 0x000a
Downgrade attack prevention	Yes, TLS_FALLBACK_SCSV supported (
SSL/TLS compression	No

Good

Good

Session resumption tickets are less good than caching

OCSP cert stapling is good

HSTS (HTTP strict transport security) modifies clients, beware

Weak-ish

Forward Secrecy	With some browsers (more info)
ALPN	Yes http/1.1
NPN	No
Session resumption (caching)	Yes
Session resumption (tickets)	Yes
OCSP stapling	No
Strict Transport Security (HSTS)	Yes max-age=31536000;includeSubDomains
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

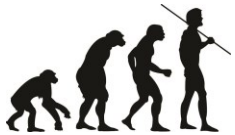
Good

Weak

(but cert offers OCSP URL)

Not good

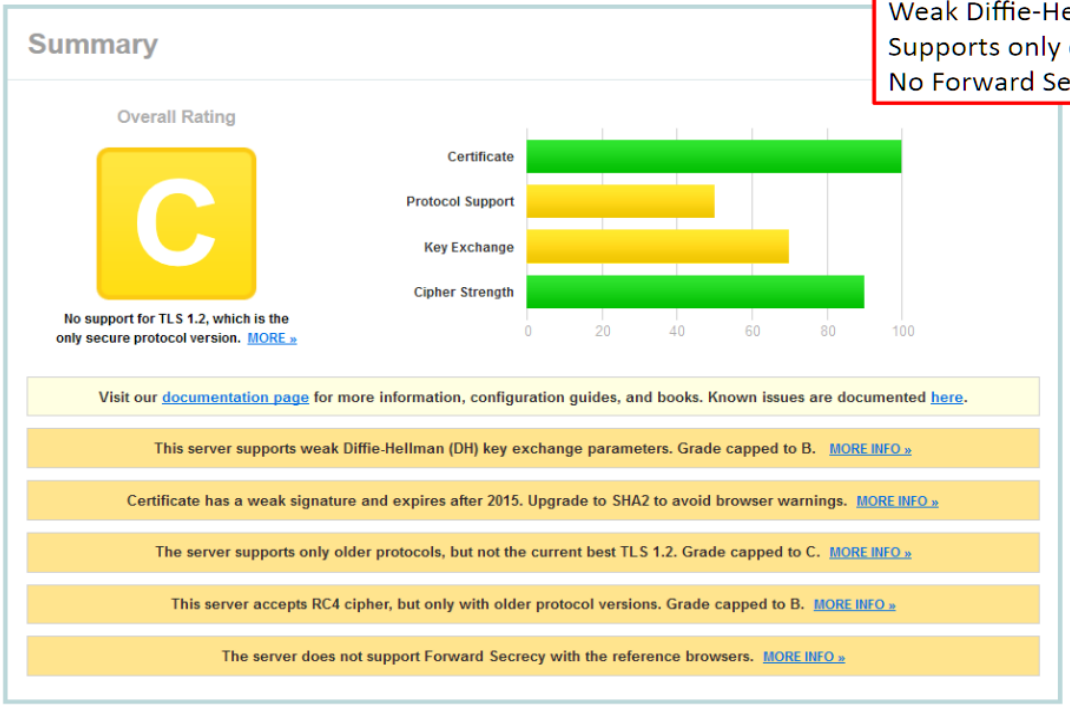
A grade C situation, evolution of TLS rules in action



Stock OES11 SP2/SLES11 SP3

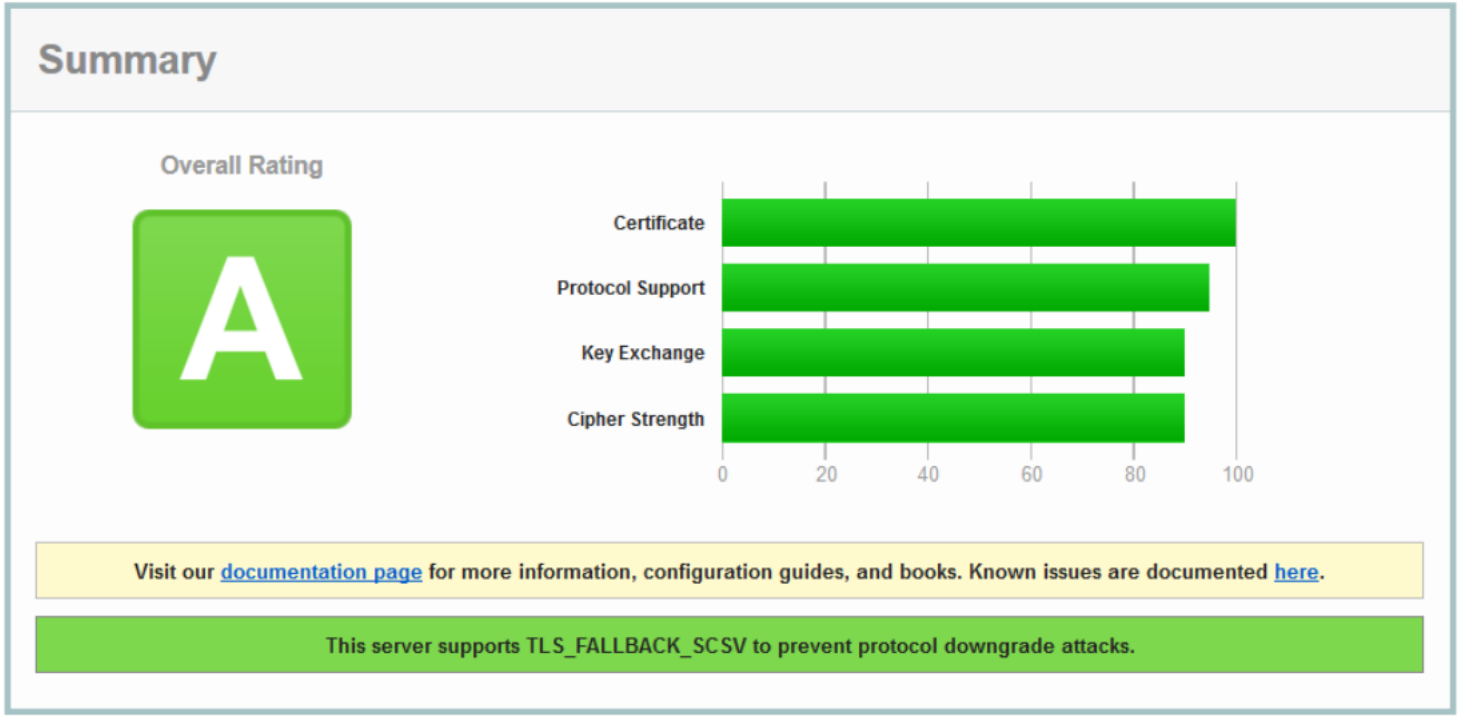
http://ssllabs.com/

RC4, No TLS v1.2
Weak Diffie-Hellman keys
Supports only older protocols
No Forward Secrecy



This faithful server was fine, but alas evolution of TLS rules has made it be down graded

OES11 SP2 after modernization steps



Object lesson: it can be done, progress can occur

From https://netlab1.net/long-term/Apache-TLSv1.2-Rev2.pdf

Internet reports about EMAIL port 25 with StartTLS

Excerpts:

https://www.checktls.com/

MTA-STS & DANE are so-so attempts at controlling TLS usage, best ignored

crypto

Pref	Answer	Connect	HELO	TLS	Cert	Secure	From	MTA-STS	DANE	Score
0	OK (91ms)	OK (91ms)	OK (90ms)	OK (91ms)	OK (1,120ms)	OK (95ms)	OK (121ms)	no policy	no TLSA	97.88
30	OK (88ms)	OK (88ms)	OK (88ms)	OK (88ms)	OK (1,336ms)	OK (91ms)	OK (91ms)	no policy	no TLSA	97.88
	100%	100%	100%	100%	100%	100%	100%			98

https://ssl-tools.net/

Priority	STARTTLS	Certificates	Protocol			
0	supported ✔	not checked	DANE ? missing		TLSv1.2	2021-04-30
			PFS ✔ supported		~~SSLv3~~	1 s
			Heartbleed ✔ not vulnerable			
			Weak ciphers ✔ not found			

There are many Internet SMTP delivery test offerings, but do be aware of limited test techniques.
Many applications (ftp, telnet, ldap etc) are relatively simplistic about TLS details (our pc^3 items).
DKIM can assist by providing encrypted checksums of message components. DNS record has the key.
See Linux application amavisd, Internet DKIM checkers, and discussions in
http://dkim.org/ https://en.Wikipedia.org/wiki/DomainKeys_Identified_Mail
https://www.cloudflare.com/engb/learning/dns/dns-records/dns-dkim-record/
https://www.mailhardener.com/kb/how-to-create-a-dkim-record-with-openssl
Comprehensive TLS information is available from local application *testssl.sh,* from https://testssl.sh.
Also see https://www.feistyduck.com/library/openssl-cookbook/online/ch-testing-with-openssl.html.

Looking into details

A useful tool and comments on some crypto controls

This part is likely to be boring to read, until we start modifying our machines…

Mozilla advisor for OES2018 SP2 Apache web

(A convenient helper, not a tester,
advisory only, use our good judgements)

moz://a

SSL Configuration Generator

Server Software

- (•) Apache
- ( ) AWS ALB
- ( ) AWS ELB
- ( ) Caddy
- ( ) Dovecot
- ( ) Exim
- ( ) Go
- ( ) HAProxy
- ( ) Jetty
- ( ) lighttpd
- ( ) MySQL
- ( ) nginx
- ( ) Oracle HTTP
- ( ) Postfix
- ( ) PostgreSQL
- ( ) ProFTPD
- ( ) Redis
- ( ) Tomcat
- ( ) Traefik

Mozilla Configuration

- ( ) Modern
  Services with clients that support TLS 1.3 and don't need backward compatibility
- (•) Intermediate
  General-purpose servers with a variety of clients, recommended for almost all systems
- ( ) Old
  Compatible with a number of very old clients, and should be used only as a last resort

Environment

Server Version	2.4.23
OpenSSL Version	1.0.2p

Miscellaneous

- [ ] HTTP Strict Transport Security
  This also redirects to HTTPS, if possible
- [x] OCSP Stapling

From https://ssl-config.mozilla.org
Also see its Resources section for discussion

Mozilla advisor for OES2018 SP2 Apache web

```
# this configuration requires mod_ssl and mod_socache_shmcb
<VirtualHost *:443>
    SSLEngine on

    # curl https://ssl-config.mozilla.org/ffdhe2048.txt >> /path/to
/signed_cert_and_intermediate_certs_and_dhparams
    SSLCertificateFile      /path/to/signed_cert_and_intermediate_certs_and_dhparams
    SSLCertificateKeyFile   /path/to/private_key

    # enable HTTP/2, if available
    Protocols h2 http/1.1
</VirtualHost>

# intermediate configuration
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite          ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
SSLHonorCipherOrder     off
SSLSessionTickets       off

SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
```

Copy

This report is most useful for suggesting SSL Cipher Suite names (as one line)

SSLHonorCipherOrder should be ***on*** to have server choose cipher suites

Better is use app's recommendation

Worth remembering: we should think about details, not just copy & paste.

Items to note:

SSLProtocol
SSLCiperSuite
SSLHonorCipherOrder
SSLSessionTickets
SSLUseStapling
SSLStaplingCache

Mozilla advisor for Tomcat v9 (used in OES2018)

```
<Connector
    port="443"
    SSLEnabled="true">

    <!-- TLS 1.3 requires Java 11 or higher -->
    <SSLHostConfig
        ciphers="ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384"
        disableSessionTickets="true"
        honorCipherOrder="false"
        protocols="TLSv1.2">

        <Certificate
            certificateFile="/path/to/signed_certificate"
            certificateChainFile="/path/to/intermediate_certificate"
            certificateKeyFile="/path/to/private_key" />
    </SSLHostConfig>

    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
```

Copy

What to do with this in OES is presently a puzzle

What is in a cipher suite name? I thought you might never ask.

`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`

TLS v1.2 cipher suite name components (The IANA uses prefix TLS_, openssl does not)

Key Exchange Algorithms (RSA, DH, ECDH, DHE, ECDHE, PSK)	Let's exchange keys
Authentication/Digital Signature Algorithm (RSA, ECDSA, DSA)	A note from my mother, challenge it at your peril
WITH	TLS v1.3 shortens the name to just the next two items
Bulk Encryption Algorithms (AES, CHACHA20, Camellia, ARIA)	Crypto for user's data
Message Authentication Code Algorithms (SHA-256, POLY1305)	Crypto for data checksum

"Oh yes of course, that makes everything be crystal clear. Now then, where were we?"

From https://www.thesslstore.com/blog/cipher-suites-algorithms-security-settings/

HSTS, best to avoid this fumble

Modifies clients, an unwise action

HSTS mechanism overview [edit]

from Wikipedia.org

A server implements an HSTS policy by supplying a header over an HTTPS connection (HSTS headers over HTTP are ignored).[1] For example, a server could send a header such that future requests to the domain for the next year (max-age is specified in seconds; 31,536,000 is equal to one non-leap year) use only HTTPS: `Strict-Transport-Security: max-age=31536000`.

When a web application issues HSTS Policy to user agents, conformant user agents behave as follows (RFC 6797):[9]

1. Automatically turn any insecure links referencing the web application into secure links (e.g. `http://example.com/some/page/` will be modified to `https://example.com/some/page/` *before* accessing the server).
2. If the security of the connection cannot be ensured (e.g. the server's TLS certificate is not trusted), the user agent must terminate the connection (RFC 6797 section 8.4, Errors in Secure Transport Establishment) and should not allow the user to access the web application (section 12.1, No User Recourse).

My views: This asks clients to rewrite the server's web page, but only clients willing to play this game. Silly.

Servers should control affairs. Clients can/will do as they wish. Asking is not being strict, yet invading is offensive and has legal aspects.

From https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security:

> **Note:** The `Strict-Transport-Security` header is **ignored** by the browser when your site is accessed using HTTP; this is because an attacker may intercept HTTP connections and inject the header or remove it. When your site is accessed over HTTPS with no certificate errors, the browser knows your site is HTTPS capable and will honor the `Strict-Transport-Security` header.

I **redirect** all incoming *http* traffic to *https*. In file /etc/apache2/default-server.conf, or in /etc/apache2/global.conf if it is enabled, are placed these three directives:

```
RewriteEngine  on
RewriteCond  %{HTTPS}  off
RewriteRule  (.*)  https://%{HTTP_HOST}%{REQUEST_URI}  [QSA,L,R=301]
```

If the request did not use https then do the rule

Query String Append

Redirect

This server-side redirect cleanly ensures https usage. No client modifications.

A better approach

HSTS is **invasive**. Think very carefully before deploying it. Clear browser HSTS: https://www.thesslstore.com/blog/clear-hsts-settings-chrome-firefox/ and others.

Aside: Apache Rewrite has security benefits

The http ***redirect*** example on the previous slide turns out to rebuff a large number of penetration attempts because those remote programs, unlike normal web browsers, seem unable to change from http to https. I also added statement RewriteBase *mydocroot* within the default server (port 80) clause. *Observe redirect work on your systems (*review Apache's access & error logs).

Hint: in file /etc/apache2/httpd.conf remove <IfDefine brackets around including global.conf, and ensure "ssl" is cited early (~3rd) in line APACHE_MODULES in file /etc/sysconfig/apache2.

Another example is rejecting requests using ancient weak form HTTP/1.0 (vs 1.1 & 2.0 of today) as well as unwanted requests OPTIONS and CONNECT. Each Condition detects an unwanted request kind and OR's its result with that of the next test, so that any of them can produce a Fail response.

```
RewriteEngine On
RewriteCond %{THE_REQUEST}  ^OPTIONS  [OR]
RewriteCond %{THE_REQUEST}  ^CONNECT  [OR]
RewriteCond %{THE_REQUEST}  HTTP/1\.0$
RewriteRule  ".*"  "-"  [F]
```

request starts with OPTIONS
request starts with CONNECT
request ends with HTTP/1.0
No ("-") returned request text

These five commands would best be placed before the ***redirect*** triad. I recommend avoiding HSTS.

A pleasant case is politely saying that a requested item is Gone, not here any more, out of stock:

```
RewriteEngine On
RewriteRule  "/pub/goodies/*"  "-"  [G,NC]
```

The Apache manual and Internet comments have further discussion and examples about this topic.

Ordering of tests, based on observing logs

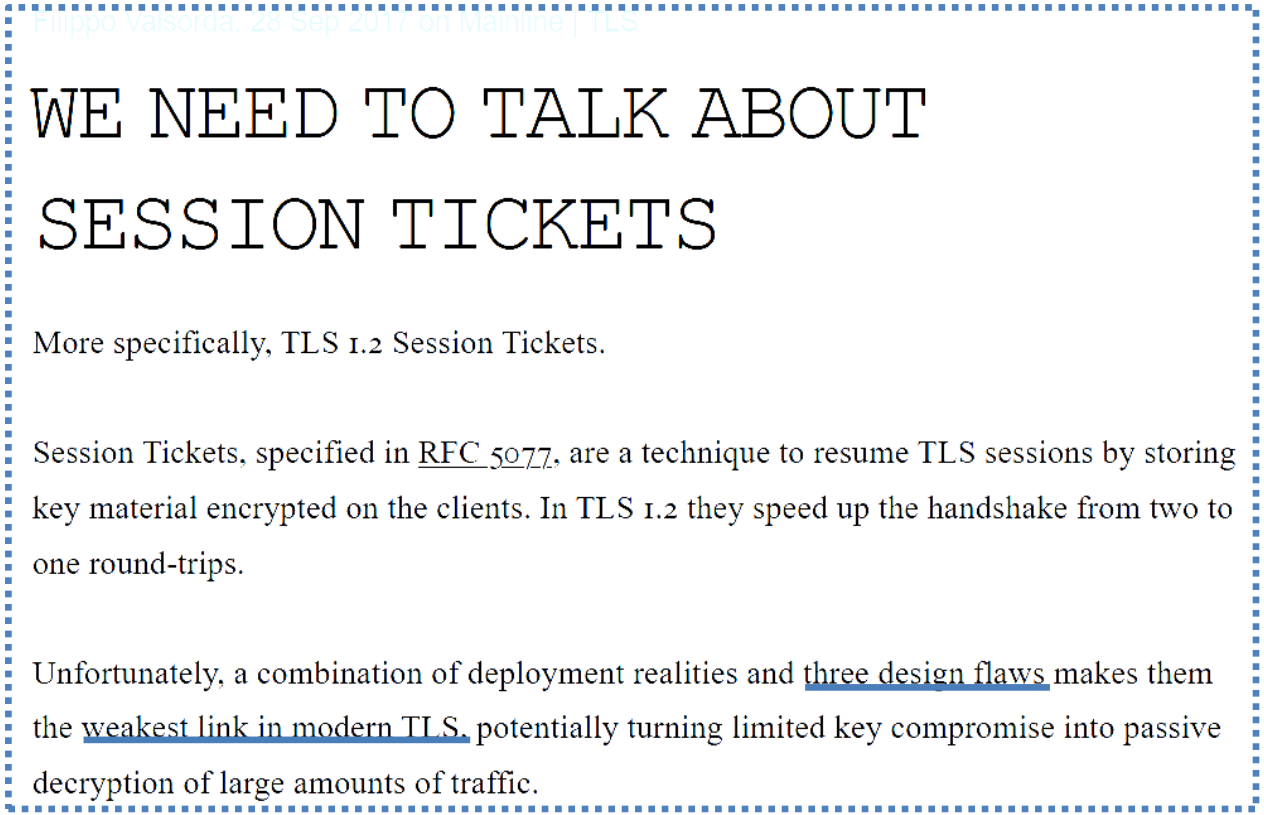
For more details on this item please have a look at presentation Configuring Apache web server Apache config. Also note that an effective ordering of firewall filter rules for Apache is first do the redirection of http to https, then reject requests using http v1.0 (vulnerable), and follow that with a list of web request filter rules. Real web clients transparently follow redirect commands.

```
<IfModule mod_rewrite.c>
# Fail OPTIONS, CONNECT etc and HTTP/1.0 requests, and require https://
# ^ is "starts with", $ is "ends with“
RewriteEngine On          #Please repeat the request using https, not http. This rebuffs about 90% of script idiot attempts
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}/%{REQUEST_URL} [QSA,L,R=301]      #R is redirect, args are repeated

RewriteEngine On
RewriteCond %{THE_REQUEST} HTTP/1\.0$                    # HTTP/1.0 is old and vulnerable to abuse
RewriteRule ".*" "-" [F,NC,L,END]                        # F is report failure, L is last, END is end this rule set

# Rebuff  the more intelligent penetration attempts
RewriteEngine On
RewriteCond %{REQUEST_METHOD} ^(CONNECT|DELETE|HEAD|OPTIONS|PUT|TRACE|TRACK)
RewriteRule ".*" "-" [F,NC,L,END]                        #  “.*” “-”  is replace incoming command text with -
<IfModule>
```

TLS Session Resumption tickets considered weak. Prefer cache



Filippo Valsorda, 28 Sep 2017, on Mailing | TLS

WE NEED TO TALK ABOUT SESSION TICKETS

More specifically, TLS 1.2 Session Tickets.

Session Tickets, specified in RFC 5077, are a technique to resume TLS sessions by storing key material encrypted on the clients. In TLS 1.2 they speed up the handshake from two to one round-trips.

Unfortunately, a combination of deployment realities and three design flaws makes them the weakest link in modern TLS, potentially turning limited key compromise into passive decryption of large amounts of traffic.

Excerpt:
“As soon as a key requires distribution it's exposed to an array of possible attacks that an ephemeral key in memory doesn't face.”

Worth reading. TLS v1.3 tries to address this problem.

From https://blog.filippo.io/we-need-to-talk-about-session-tickets/
Also see https://upb-syssec.github.io/blog/2023/session-tickets/

OES TID: avoiding difficulties from Certificate Revocation Lists

Disabling CRL verification tree-wide in eDirectory 8.8.x, 9.0 & 9.1 Certificate Server

- **Document ID:***7022461*
- **Creation Date:**13-Dec-2017
- **Modified Date:**10-Jul-2018
- **Micro Focus Products:**
  - eDirectory

Environment

eDirectory 9.1
eDirectory 9.0.4
eDirectory 8.8.8.11
iManager 3.1
iManager 3.0.4
iManager 2.7 SP7
NetIQ Certificate Server

Situation

Joining a new server to the tree is taking over two hours.

Validating public keys on certificates returns: Invalid: CRL Decode Error

Routers and Firewalls are strictly filtering out unsecure ports such as port 80 and 8028.

Browsers are complaining about being unable to get the CRL (Certificate Revocation List) using the certificate's CDP (Certificate Distribution Point) URL.

From https://support.microfocus.com/kb/doc.php?id=7022461#

CRLs can also influence/hinder OES migrations

See following OCSP feature as a better method

On-line Cert Status Protocol: OCSP Certificate Stapling

As of 2017-10, **No**.

Dovecot does not have any OCSP support whatsoever, as of 2016 was considering the feature for a future release, no work has been done on that since.

Postfix does not have any OCSP support whatsoever, and as of 2017 is not planning to ever to *ever* implement such feature.

Exim can provide clients with an OCSP response, yet acquiring such is yet left as an exercise to the admin.

The main arguments against adding such support are:

1. Security features should be simple so they have more benefit than added risks. OCSP is complex. Short certificate validity is simple and mitigates the same issue.
2. The Chicken-Egg problem of OCSP support in servers being entirely useless until MUAs add such support.

This does not hinder the usage of `must-staple` certificates in web servers. Just have the option enabled on your web server certificate (e.g. `www.example.com`) and disabled on your mail server certificate (e.g. `mail1.example.com`).

Stapling: Periodically query the CA about cert validity, and cache results. Even if the certificate does state an OCSP responder URL our web server can supply the results within the TLS handshake. That is good. This is a TLS extension in RFC 6066. Also see your Apache2 manual about *mod_ssl* for OCSP details.

Stapling avoids Cert Revocation List complications.

Web servers may support local Stapling, but Postfix, Dovecot, LDAPs, SSHd and so on likely do not (they lack an ability to talk to the CA). Beware *Must Staple*. See https://scotthelme.co.uk/ocsp-must-staple/

from ssllabs.com

OCSP Must Staple	No
Revocation information	CRL, OCSP CRL: http://crl.starfieldtech.com/sfig2s1-187.crl OCSP: http://ocsp.starfieldtech.com/
Revocation status	Good (not revoked)

From https://serverfault.com/questions/830434/do-postfix-and-dovecot-support-ocsp-stapling

Also see https://raymii.org/s/articles/OpenSSL_Manually_Verify_a_certificate_against_an_OCSP.html

Tomcat OCSP support advice, the devil is in the <many> details

"To use Online Certificate Status Protocol (OCSP) with Apache Tomcat, ensure you have downloaded, installed, and configured the
https://tomcat.apache.org/download-native.cgi
Tomcat Native Connector.

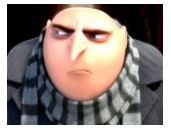
Furthermore, if you use the Windows platform, ensure you download the ocsp-enabled connector.

To use OCSP, you require the following:
OCSP-enabled certificates
Tomcat with SSL APR connector
Configured OCSP responder"

Another approach is use Apache web server as a proxy in front of Tomcat, then Apache would handle external SSL termination work.

From https://tomcat.apache.org/tomcat-9.0-doc/ssl-howto.html#SSL_and_Tomcat
See also: https://openjdk.java.net/jeps/249

Compression of HTTP content within TLS considered risky 1/2



From https://www.acunetix.com/vulnerabilities/web/crime-ssl-tls-attack/:

“CRIME is a client-side attack, but the server can protect the client by refusing to use the feature combinations which can be attacked. For CRIME, the weakness is Deflate compression. This alert is issued if the server accepts Deflate compression.

Remediation

CRIME can be defeated by preventing the use of compression, either at the client end, by the browser disabling the compression of HTTPS requests, or by the website preventing the use of data compression on such transactions using the protocol negotiation features of the TLS protocol. As detailed in The Transport Layer Security (TLS) Protocol Version 1.2, the client sends a list of compression algorithms in its ClientHello message, and the server picks one of them and sends it back in its ServerHello message. The server can only choose a compression method the client has offered, so if the client only offers 'none' (no compression), the data will not be compressed. Similarly, since 'no compression' must be allowed by all TLS clients, a server can always refuse to use compression.”

The bottom line here is **we should turn off compression by TLS itself**.

Compression within an HTTP payload is considered next.

Compression of HTTP content within TLS considered risky 2/2

SECURITY BREACH

From https://en.wikipedia.org/wiki/BREACH:

"BREACH is an instance of the CRIME attack against HTTP compression—the use of gzip or DEFLATE data compression algorithms via the content-encoding option within HTTP by many web browsers and servers.[2] Given this compression oracle, the rest of the BREACH attack follows the same general lines as the CRIME exploit, by performing an initial blind brute-force search to guess a few bytes, followed by divide-and-conquer search to expand a correct guess to an arbitrarily large amount of content."

Plus lengthy discussions in https://en.wikipedia.org/wiki/HTTP_compression

and https://security.stackexchange.com/questions/20406/is-http-compression-safe

and https://blog.qualys.com/product-tech/2013/08/07/defending-against-the-breach-attack

and https://silo.tips/download/https-secure-http which has detailed expositions

---

In my opinion this topic is confusingrly described. The clue is message length can act as a change indicator **if user input is carried** within. User initiated change can reveal encryption details.

For Apache module ***deflate*** we should restrict file types to a few kinds: not *text/plain, text/html* nor similar user input carriers, but allow non-interactive file types *image/png, video/mp4* and the like. Transfers of static *.gz* and *.zip* files do work normally, thank goodness.

Application configuration --- feeding the animals

Examples for Apache, Postfix, Dovecot, OES LDAP

These are for OES2018 SP2 and are applicable generally

Goals are

- Choose desirable protocols, ciphers and controls
- Server’s preferences govern choice of protocol and cipher
- Efficient session resumption and certificate verification (OCSP)

Apache2 create file /etc/apache2/conf.d/staple.conf, or place commands in vhost-ssl.conf or in its include file ssl-global.conf

SSLStrictSNIVHostCheck on	Require host name in vhost arrivals
SSLInsecureRenegotiation off	Default. Don't haggle in public
SSLSessionTickets off	Resumption: use cache, not tickets
SSLSessionCache shmcb:/var/lib/apache2/ssl_scache(512000)	Its cache
SSLUseStapling on	Desirable: offer local cert stapling
SSLStaplingCache shmcb:/var/run/ocsp(12800)	Cache for optional local cert stapling

shmcb: is memory caching using Apache2 module ***socache_shmcb***

SSLRandomSeed startup "file:/dev/urandom" 1024	Better random number generator

See Apache docs. SLES scatters controls within *Include* files in /etc/apache2. Double check them, particularly *global.conf* and *ssl-global.conf*. Global.conf needs to be enabled in /etc/apache2/httpd.conf by removing <IfDefine brackets around including global.conf. See also https://www.digitalocean.com/community/tutorials/how-to-configure-ocsp-stapling-on-apache-and-nginx and Apache web server documentation.

Apache2 in file vhost-ssl.conf or in its include file ssl-global.conf

# Enable/Disable SSL for this virtual host.	Each vhost can have its own cert
SSLEngine on	and SSL/TLS details
SSLProxyEngine on	
SSLCompression off	Default. Avoid TLS compression
SSLProtocol All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1	TLSv1 & TLSv1.1 are depreciated
SSLHonorCipherOrder on	Use server's order of ciphers

From Mozilla Advisor and check for weak cyphers using SSL Labs. Must be one long line:

SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384

Yes, that SSLCipherSuite line is a challenge. Consider copy&paste from Mozilla Advisor.
SLES prefers to have these items in file *ssl-global.conf*. Use your best judgement.
Test OCSP results via command **openssl s_client -connect my.host:443 -status**
and review section "OCSP Response Data:" particularly line "OCSP Response Status:"
Also "ssl" needs to be 3rd or so in line APACHE_MODULES in file /etc/sysconfig/apache2.

Postfix file main.cf

This matter goes on and on in Postfix docs. These are suggestions.

Start cipher suite selection

preempt_cipherlist, yes=Postfix chooses here, no=openssl chooses

tls_preempt_cipherlist = yes

tls_random_source = dev:/dev/urandom

tls_ssl_options = NO_COMPRESSION, NO_TICKET No compression by TLS, no ticket

From suggestion by Mozilla advisor, must be one long line. Overrides default medium cipherlist

tls_medium_cipherlist = ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256: DHE-RSA-AES256-GCM-SHA384

Email header: Received: from a.host.com (a.host.com [11.22.33.44])
(using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits))
(No client certificate requested)

See documentation at https://www.postfix.org/TLS_README.html/

See also https://kruyt.org/postfix-and-tls-encryption/ about client session renegotiation & more

Postfix file main.cf


```
# for outgoing traffic, use smtp_tls_
smtp_tls_security_level = may                                   Allow plaintext and StartTLS
smtp_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1           For "may"
smtp_tls_mandatory_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1 For mandatory TLS use
smtp_tls_mandatory_ciphers = medium
smtp_tls_session_cache_timeout = 3600s
smtp_tls_session_cache_database btree:/var/lib/postfix/smtp_tls_session_cache

# for incoming traffic, use smtpd_tls_
Same six items as above but spelled with "smtpd_tls_" rather than "smtp_tls_"
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth          ← At my place Dovecot handles SASL authentication
smtpd_sasl_security_options = noanonymous
```

Note: *postconf* displays current settings, and *postconf –d* displays defaults.
See also *testssl.sh --mx myhost* (discussed later) for detailed TLS testing.

Dovecot (IMAP4) file 10-ssl.conf

```
# SSL/TLS support: yes, no, required. <doc/wiki/SSL.txt>
ssl = yes
ssl_options = no_compression
# SSL protocols to use
##OLD ssl_protocols = !SSLv2, !SSLv3
ssl_min_protocol = TLSv1.2
# Prefer the server's order of ciphers over client's.
ssl_prefer_server_ciphers = yes
##OLD ssl_cipher_list = ALL:!LOW:!SSLv2:!EXP:!aNULL
# From suggestion by Mozilla advisor, must be one long line:
ssl_cipher_list = ECDHE-ECDSA-AES128-GM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-
SHA254:DHE-RSA-AES256-GCM-SHA384
```

Dovecot listens on
IMAP ports 143 and 993
POP3 ports 110 and 995

We see the historical change from simple cipher class descriptions to lengthy names.
See documentation at https://dovecot.org/

SSH server sshd (remains an island about its configuration)

Cryptographic policy

Symmetric algorithms for encrypting the bulk of transferred data are configured using the `Ciphers` option. A good value is `aes128-ctr,aes192-ctr,aes256-ctr`. This should also provide good interoperability.

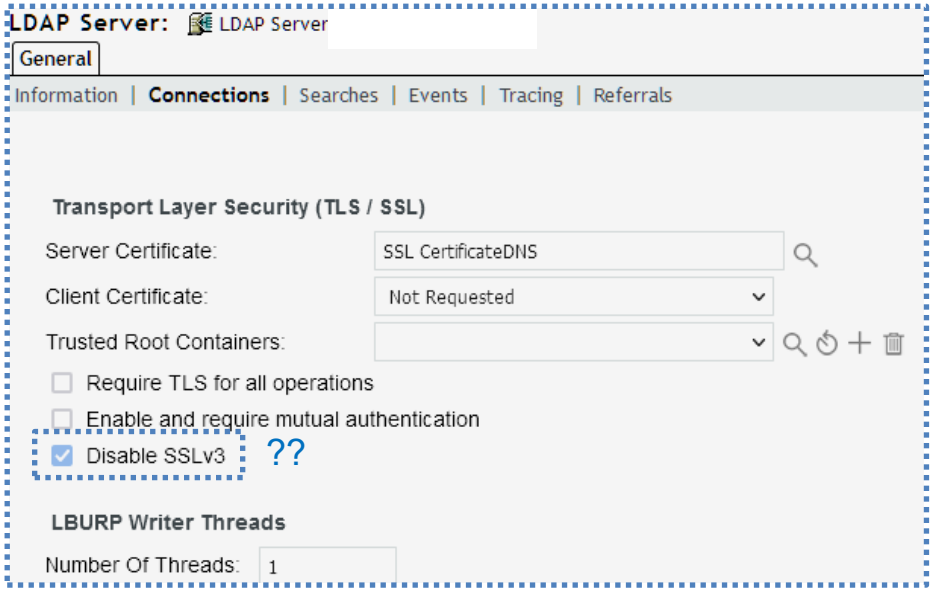
Host key algorithms are selected by the `HostKeyAlgorithms` option. A good value is `ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-rsa,ss/ssh/sshd_config/h-dss`.

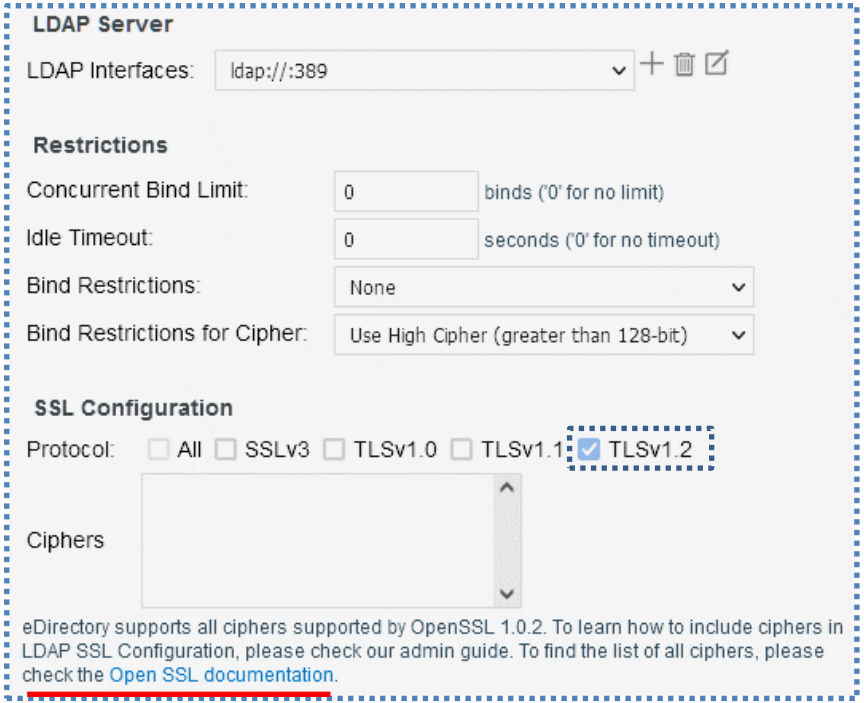
Key exchange algorithms are selected by the `KexAlgorithms` option. We recommend `ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group14-sha1,diffie-hellman-group-exchange-sha256`. In particular, we do not recommend allowing `diffie-hellman-group1-sha1`, unless needed for compatibility. It uses a 768 bit prime number, which is too small by today's standards and may be breakable by intelligence agencies in real time. Using it could expose connections to **man-in-the-middle attacks** when faced with such adversaries.

Message authentication code algorithms are configured using the `MACs` option. A good value is `hmac-sha2-256,hmac-sha2-512,hmac-sha1`.

From https://www.ssh.com/academy/ssh/sshd_config#cryptographic-policy
Also see https://infosec.mozilla.org/guidelines/openssh.html and man sshd_config

OES2018 LDAP Server controls via iManager



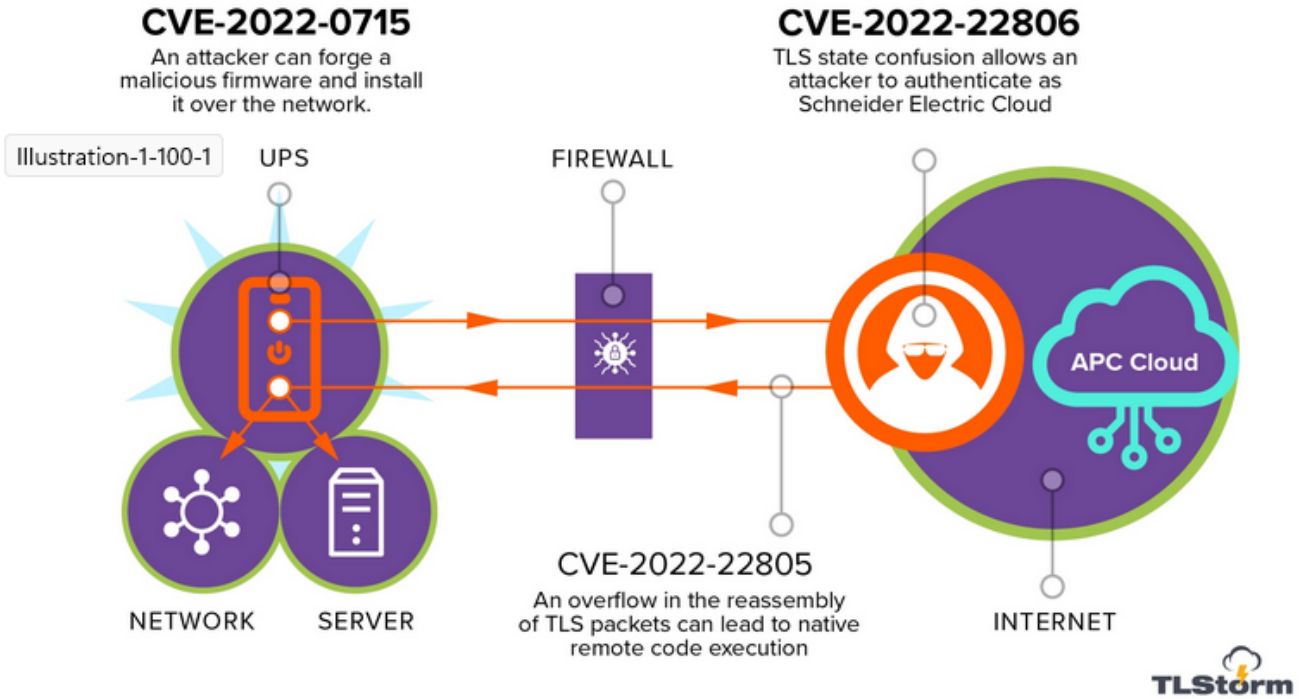


Ciphers is a jungle of complex names. Controls?
This facility and its docs need modernization.

See also https://www.netiq.com/documentation/edirectory-9/edir_admin/data/b1i4rmmx.html about configuring eDirectory for Suite B mode.

Switch and smart UPS TLS library design problems *TLStorm 2.0*

Article https://www.armis.com/research/tlstorm/ describes three serious problems in the NanoSSL library used by many popular switches and smart UPS's which present zero-click vulnerabilities.



From https://www.armis.com/research/tlstorm

“Papers please” Specifying a certificate chain, a quick note

Verifying an application’s certificate involves the machine’s (“our”) certificate plus certs of intermediary authorities which act as agents for the trusted top level CA. Thus often there is a *chain* of certificates to be supplied and verified.

Many applications ask about a *single* chain file. Create it as concatenation of the intermediary certs, starting with that closest to us, followed one by one up the chain toward the top CA. In this manner $ cat a b c >> chainfile

Some applications *do not req*uest a chain. Create a concatenation which starts with our own cert then appends the intermediaries one by one. This bulky text file is then offered to the app as “our cert”.

Also note the OSCP business verifies a chain to a result distribution URL without clients needing to verify each item. OSCP is a beneficial feature.

Many Internet test programs can display certificate chain details and health.

Advice from an expert government agency

National Security Agency | Cybersecurity Information

Eliminating Obsolete Transport Layer Security (TLS) Protocol Configurations

Executive summary

The National Security Agency (NSA) emphatically recommends replacing obsolete protocol configurations with ones that utilize strong encryption and authentication to protect all sensitive information. Over time, new attacks against Transport Layer Security (TLS) and the algorithms it uses have been discovered. Network connections employing obsolete protocols are at an elevated risk of exploitation by adversaries.

"Organizations encrypt network traffic to protect data in transit. However, using obsolete TLS configurations provides a false sense of security since it looks like the data is protected, even though it really is not. Make a plan to weed out obsolete TLS configurations in the environment by detecting, remediating, and then blocking obsolete TLS versions, cipher suites, and finally key exchange methods. Prepare for cryptographic agility to always stay ahead of malicious actors' abilities and protect important information."

From https://media.defense.gov/2021/Jan/05/2002560140/-1/-1/0/ELIMINATING_OBSOLETE_TLS_UOO197443-20.PDF
This has readable details and can be useful in discussions with higher management.
Thanks to Simon Palmer for indicating it, and to both Simon and Diana Osborn for overall reviews.

SSL/TLS scanner collection cited in the NSA document

“Scanning Tools

Comprehensive analysis of servers can be performed by attempting to initiate weak TLS sessions using custom tools and seeing if the server agrees to utilize obsolete cryptography. There are a number of open source tools and commercial services available that can perform active scans to detect non-compliant TLS versions, cipher suites, and key exchanges. The following example tools claim to be able to scan for obsolete cryptography.

https://github.com/18F/domain-scan - a scanner from GSA 18F to orchestrate scanning tools at scale. Can use the https://github.com/nabla-c0d3/sslyze Python package to scan for and report use of obsolete cryptography.

https://pentest-tools.com/network-vulnerability-scanning/ssl-tls-scanner

- https://www.ssllabs.com/ssltest

https://testtls.com/ - allows scanning any TCP port, both on IPv4 and IPv6

https://gf.dev/tls-scanner

https://github.com/prbinu/tls-scan

https://www.thesslstore.com/ssltools/ssl-checker.php

- used here

https://www.wormly.com/test_ssl

https://www.digicert.com/help/

https://www.hardenize.com

https://www.tenable.com/plugins/was/families/SSL%2FTLS for use with Tenable software.

- https://github.com/drwetter/testssl.sh

https://github.com/rbsec/sslscan - a feature-rich command line SSL/TLS scanner with color-coded output; works on Windows, MacOS, and Linux.”

From https://github.com/nsacyber/Mitigating-Obsolete-TLS

Locally run script testssl.sh as *./testssl.sh hostname*

```
###########################################################
    testssl.sh       3.0 from https://testssl.sh/

      This program is free software. Distribution and
             modification under GPLv2 permitted.
      USAGE w/o ANY WARRANTY. USE IT AT YOUR OWN RISK!

       Please file bugs @ https://testssl.sh/bugs/

###########################################################

 Using "OpenSSL 1.0.2-chacha (1.0.2k-dev)" [~183 ciphers]
 on netlab:./bin/openssl.Linux.x86_64
 (built: "Jan 18 17:12:17 2019", platform: "linux-x86_64")


 Start 2021-05-18 14:12:06

 rDNS
 Service detected:       HTTP


 Testing protocols via sockets except NPN+ALPN

 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    offered (OK)
 TLS 1.3    not offered and downgraded to a weaker protocol
 NPN/SPDY   not offered
 ALPN/HTTP2 h2, http/1.1 (offered)
```

Testing OES2018 SP2 Apache.
Four screens follow.

This small script is a useful tool, similar to that of SSL Labs, and it can reveal other useful detail.

It can test more than web serving.

```
testssl.sh -t smtp host:25
testssl.sh -t imap host:143
testssl.sh --mx host
testssl.sh host:636
```

plus -t ftp, lmtp, xmpp, telnet, ldap, etc.
-t means try StartTLS with the protocol.
See testssl.sh --help for full listing.
To use your local openssl add option

```
--openssl /usr/bin/openssl
```

See https://testssl.sh

Testssl.sh

```
 Testing cipher categories

 NULL ciphers (no encryption)                not offered (OK)
 Anonymous NULL Ciphers (no authentication)  not offered (OK)
 Export ciphers (w/o ADH+NULL)               not offered (OK)
 LOW: 64 Bit + DES, RC[2,4] (w/o export)     not offered (OK)
 Triple DES Ciphers / IDEA                   not offered
 Obsolete: SEED + 128+256 Bit CBC cipher     not offered
 Strong encryption (AEAD ciphers)            offered (OK)


 Testing robust (perfect) forward secrecy, (P)FS -- omitting Null Authentication/Encryption, 3DES, RC4

 PFS is offered (OK)          ECDHE-RSA-AES256-GCM-SHA384 DHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES128-GCM-SHA256
                              DHE-RSA-AES128-GCM-SHA256
 Elliptic curves offered:     secp256k1 prime256v1 secp384r1 secp521r1 brainpoolP256r1 brainpoolP384r1
                              brainpoolP512r1
 DH group offered:            RFC3526/Oakley Group 14 (2048 bits)

 Testing server preferences

 Has server cipher order?     yes (OK)
 Negotiated protocol          TLSv1.2
 Negotiated cipher            ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Cipher order
    TLSv1.2:   ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES256-GCM-SHA384 DHE-RSA-AES128-GCM-SHA256
               DHE-RSA-AES256-GCM-SHA384
```

- Noted protocols, ciphers, controls and certificate pc[3]

Testssl.sh

We recall “long goodbyes”. Here is a “long hello”.

Hello

```
 Testing server defaults (Server Hello)

 TLS extensions (standard)     "server name/#0" "renegotiation info/#65281" "EC point formats/#11"
                               "status request/#5" "heartbeat/#15" "application layer protocol negotiation/#16"
 Session Ticket RFC 5077 hint  no -- no lifetime advertised
 SSL Session ID support        yes
 Session Resumption            Tickets no, ID: yes
 TLS clock skew                Random values, no fingerprinting possible
 Signature Algorithm           SHA256 with RSA
 Server key size               RSA 2048 bits
 Server key usage              Digital Signature, Key Encipherment
 Server extended key usage     TLS Web Server Authentication, TLS Web Client Authentication
 Serial / Fingerprints         31610444D74273BA / SHA1 48DA8E8B6F5DEAC46ADC19AFCCED02D1E268A1AD
                               SHA256 91D2D1E0BECDD2BB60A0388BFD6937902ED9EEE1D0FD9FCA45A81B879B00EE8C

 Common Name (CN)              *.netlab1.net
 subjectAltName (SAN)          *.netlab1.net netlab1.net
 Issuer                        Starfield Secure Certificate Authority - G2 (Starfield Technologies, Inc. from US)
 Trust (hostname)              Ok via SAN (same w/o SNI)
 Chain of trust                Ok
 EV cert (experimental)        no
 ETS/"eTLS", visibility info   not present
 Certificate Validity (UTC)    291 >= 60 days (2020-03-24 03:38 --> 2022-04-07 08:00)
 # of certificates provided    3
 Certificate Revocation List   http://crl.starfieldtech.com/sfig2s1-187.crl
 OCSP URI                      http://ocsp.starfieldtech.com/
 OCSP stapling                 offered, not revoked
 OCSP must staple extension    --
 DNS CAA RR (experimental)     not offered
 Certificate Transparency      yes (certificate extension)
```

Details are in the URL below, if interested

Server name is same as in the cert
Intermediary chain is valid

Cert offers OCSP responder URL,
Apache also offers OCSP results

Extensions: https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml

Testssl.sh

```
 Testing vulnerabilities

 Heartbleed (CVE-2014-0160)                not vulnerable (OK), timed out
 CCS (CVE-2014-0224)                       not vulnerable (OK)
 Ticketbleed (CVE-2016-9244), experiment.  not vulnerable (OK), no session ticket extension
 ROBOT                                     Server does not support any cipher suites that use RSA key trans
port
 Secure Renegotiation (RFC 5746)           supported (OK)
 Secure Client-Initiated Renegotiation     not vulnerable (OK)
 CRIME, TLS (CVE-2012-4929)                not vulnerable (OK)
 BREACH (CVE-2013-3587)                    no HTTP compression (OK)  - only supplied "/" tested
 POODLE, SSL (CVE-2014-3566)               not vulnerable (OK), no SSLv3 support
 TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible (OK), no protocol below TLS 1.2 offered
 SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
 FREAK (CVE-2015-0204)                     not vulnerable (OK)
 DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on this host and port (OK)
                                           make sure you don't use this certificate elsewhere with SSLv2 en
abled services

                                           https://censys.io/ipv4?q=91D2D1E0BECDD2BB60A0388BFD6937902ED9EEE
1D0FD9FCA45A81B879B00EE8C could help you to find out
 LOGJAM (CVE-2015-4000), experimental      common prime with 2048 bits detected: RFC3526/Oakley Group 14 (2
048 bits),

                                           but no DH EXPORT ciphers
 BEAST (CVE-2011-3389)                     not vulnerable (OK), no SSL3 or TLS1
 LUCKY13 (CVE-2013-0169), experimental     not vulnerable (OK)
 RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers detected (OK)
```

Vulnerabilities see: https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/

Testssl.sh

```
 Testing 370 ciphers via OpenSSL plus sockets against the server, ordered by encryption strength

Hexcode  Cipher Suite Name (OpenSSL)       KeyExch.   Encryption  Bits     Cipher Suite Name (IANA/RFC)
------------------------------------------------------------------------------------------------------------------------
 xc030   ECDHE-RSA-AES256-GCM-SHA384       ECDH 256   AESGCM      256      TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
 x9f     DHE-RSA-AES256-GCM-SHA384         DH 2048    AESGCM      256      TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
 xc02f   ECDHE-RSA-AES128-GCM-SHA256       ECDH 256   AESGCM      128      TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
 x9e     DHE-RSA-AES128-GCM-SHA256         DH 2048    AESGCM      128      TLS_DHE_RSA_WITH_AES_128_GCM_SHA256


 Running client simulations (HTTP) via sockets

 Android 4.4.2                TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 5.0.0                TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 6.0                  TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 7.0                  TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 8.1 (native)         TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 9.0 (native)         TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Android 10.0 (native)        TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Chrome 74 (Win 10)           TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Chrome 79 (Win 10)           TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Firefox 66 (Win 8.1/10)      TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Firefox 71 (Win 10)          TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 IE 6 XP                      No connection
 IE 8 Win 7                   No connection
 IE 8 XP                      No connection
 IE 11 Win 7                  TLSv1.2 DHE-RSA-AES128-GCM-SHA256, 2048 bit DH
 IE 11 Win 8.1                TLSv1.2 DHE-RSA-AES128-GCM-SHA256, 2048 bit DH
 IE 11 Win Phone 8.1          No connection
 IE 11 Win 10                 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Edge 15 Win 10               TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
 Edge 17 (Win 10)             TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
```

- Forward Secrecy

and so on

Simulations are useful.
SSL Labs report also has them.

Crypto Park souvenir and book shop

References about SSL/TLS mechanics for ordinary usage:

SSL/TLS for system admins v2 doc video v2.1 doc a summary with practical details
Testers, links to ssllabs.com testssl.sh checktls.com digicert.com
Application TLS configuration suggestions, link to ssl-config.mozilla.org
SSL/TLS links, to engine: openssl.org to discussion: feistyduck.com
Disable certificate CRL verification in eDir 8/9, link to MF doc
Certificate recreation script for OES2018/2015/11, link to MF doc
Introduction v2 to "Let's Encrypt" free certificates doc
How to move OES Cert Authority to another server, link to MF doc
OCSP Cert Stapling, links to Apache&Nginx Postfix discuss discuss
DKIM email protection, links to www.sidn.nl amavisd Wikipedia
US NSA report about eliminating obsolete TLS configurations, link to doc
US NSA Network Infrastructure Security Guidance, link to doc
Switch/UPS TLS design faults, link to TLStorm 2.0 doc
Improving for TLS v1.2 (2016): Apache v2.2 Postfix+Dovecot
Enroute TLS interceptions, local copy link to doc
Explaining terminology, links to PKI intro PKI in wikipedia SSL handshake
Elliptic Curve Crypto ECC vs RSA Cipher Block Chaining
Diffie-Hellman TLS v1.3 TLS interception SMS spoofing

Links to further tutorial/discussion articles:

The Illustrated TLS Connection Every Byte Explained (tls.ulfheim.net)
TLS Security 5: Establishing a TLS Connection (www.acunetix.com)
We need to talk about Session Tickets (blog.filippo.io)
Apache Modsecurity Handbook (feistyduck.com)
Apache security ebook (feistyduck.com)
Postfix hardening and security (linux-audit.com)
OpenSSL Cookbook (feistyduck.com)
Certificate PKI tutorial (www.cs.auckland.ac.nz)
Mapping OpenSSL cipher suite names to IANA names (testssl.sh)
How to use Wireshark to troubleshoot SSL/TLS issues (www.ssltrust.com)
Another Wireshark how-to (trickster.dev, via feistyduck.com citation)

SSL/TLS mechanics collection on netlab1.net, TTP private area

These are two screen capture images

We are back home now. Next time you conduct the tour.

SSL/TLS is important these days, as we all know, but it is complicated.

We progress by using good testing tools and examples, think and do homework reading, adjust application configurations, then re-test.

We have seen several tools plus examples. The netlab1.net TLS mechanics collection has pointers to read more about tools & items. Best is run these tests on your systems to have full reports. Then consider sharing your test results and adjustments with colleagues.

Developers and their products could also benefit from a TLS safari.

No math nor logic quiz at this point. However, the Internet does set exams for us, and grader Ms Nature is not very sympathetic.

MindWorks Inc. Ltd
210 Burnley Road
Weir
Bacup
OL13 8QE UK

Telephone: +44 (0) 170 687 1900
Fax: +44 (0) 170 687 8203
Web: www.mindworksuk.com
Email: training@mindworksuk.com