TLS/SSL hardening and compatibility Report 2011

Update to the 2010 Report

Author: Thierry ZOLLER
contact@g-sec.lu
http://www.g-sec.lu

G-SEC™ is a non-commercial and independent group of Information Security Specialists based in Luxembourg.

Table of Contents

Introduction

This report gives general recommendations as to how to configure SSL/TLS in order to provide state of the art authentication and encryption. The options offered by SSL engines grew from the early days since Netscape developed SSL2.0. The introduction of TLS made matters more challenging as **servers and clients** offer different sets of available options depending on which SSL engine (OpenSSL, NSS, SCHANNEL etc...) they use. Finding the middle ground has proven difficult especially as the supported protocols and cipher suites are mostly not documented.

To make matters more complicated Browsers may not use all functionality offered by the SSL stack, **this report will only list functionality used by current Browsers**.

This report provides an overview of the currently available TLS options across Servers and Clients and allows you to offer support for a wide variety of Browsers an offer "good enough" security.

The 2011 version was updated as follows:

- Google Chrome moved away from Microsoft SCHANNEL and now uses Network Security Services (NSS) offering high end cryptography on legacy windows systems (XP,2000).
- Added Opera Cipher and Protocol Support
- Style Errors

During the creation of this Document two Tools have been developed:

- *SSL Harden* (beta) – Allows users of Windows 2000, XP, Vista, 7 and particularly administrators of Windows Server 2003 & 2008R2 to harden SSL/TLS support. Administrators can manually edit and backup the SSL configuration and set PCI-DSS compliant SSL rules with a click of a button. Link
- *SSL Audit* (alpha) - A remote SSL audit tool able scan for SSL/TLS support against remote servers. SSL Audit uses its own small parsing engine and does not rely on OpenSSL or other SSL engines allowing it to detect ciphers not supported by OpenSSL. Link

Please note that this summary does not take into account the arrival of quantum computing. Large quantum computers able to crack large RSA keys are foreseen for 2014 by the ARDA and 2018 by Prof Lloyd [1] . Shor's algorithm could then be used to break the RSA key sizes very fast. We recommend to push for ECC based certificates as soon as possible.

The information is believed to be correct at the time of writing, due to the nature of undocumented features there might be slight errors in this version if you believe the

[1] http://synaptic-labs.com/ecosystem/context-qc-relevant-today.html

information displayed within this paper is wrong please contact contact@g-sec.lu. Feedback from Microsoft, Apache, Opera and Apple was integrated when available.

Revisions

Version	*Date*	*Annotations*
0.8	07.12.2009	Initial draft
0.85	09.12.2009	Added recommendations, Added BSI, NIST, FSIA recommendations
0.9	09.12.2009	Added Browser support Added Server support
0.95	18.12.2009	Synopsis
0.96	05.01.2010	Released for RFC
0.97	18.01.2010	Released as RC
0.98	23.01.2010	Fixed a few typos
0.99	12.03.2011	Added changes to chrome, corrected grammar.
1.0	21.09.2011	Released as 1.0
1.01	25.09.2011	Layout, added details provided by Opera
1.02	28.09.2011	Update mod_gnutls, formating

Introduction to SSL/TLS

In order to securely transport data from one endpoint to another SSL and TLS protocols are used as they provide data confidentiality and data integrity. TLS was designed to offer a flexible and secure protocol that is able to interoperate with any service or application, furthermore TLS provides cryptographic support that SSL could not offer.

SSL/TLS Protocol versions

SSLv2

SSL version 2 was developed by Netscape in 1996 and is 13 years old; it is vulnerable to various attacks and should not be supported. Internet browsers like Internet Explorer 7 (2006), Firefox 2 (2005) and Opera 9 (2006) **do no longer support SSLv2**.

Users should not be encouraged to use older browsers as they suffer from other vulnerabilities that put them at risk. Should another requirement such as third party code require SSLv2 for an e-banking platform it needs to be upgraded to TLS, as it is vulnerable to several known attacks.

Should you absolutely need to conform to foreign regulations we recommend relocating these customers to a separated banking server/system. They pose a risk for other e-banking users. (SSLv2 does not support perfect forward secrecy)

The SSLv2 protocol suffers from

- Re-usage of key material (message authentication and encryption) thus, in case of EXPORT ciphers unnecessarily weakening the MAC (not required by export restrictions)
- Ciphers marked as “Export” have an arbitrary small key size and can be cracked easily with today’s hardware.
- weak MAC construction and supports only MD5 hash function
- padding length field is unauthenticated [2]
- Downgrade attack – an attacker may downgrade the encryption to the lowest available and after doing so crack the keys.
- Truncation attacks – The attacker may reset the TCP connection and as such

[2] Analysis of the SSL 3.0 Protocol - David Wagner et al

Differences between SSLv3 and SSLv2

- Key material is no longer reused in both Message authentication and encryption making suites marked as EXPORT „stronger“.
- MAC construction enhanced and support for SHA1 added
- SSLv3 adds protection of the Handshake, server-side can detect downgrade attacks
- SSLv3 adds support for a closure alert

Differences between TLS v1and SSLv3

- Expansion of cryptographic keys from the initially exchanged secret was improved
- MAC construction mechanism modified into an HMAC
- Mandatory support for Diffie-Hellman key exchange, the Digital Signature Standard, and Triple-DES encryption

Differences between TLS v1.1 and TLS v1 [3]

- The implicit Initialization Vector (IV) is replaced with an explicit IV to protect against CBC attacks[4]
- Handling of padding errors is changed to use the bad_record_mac
- Alert rather than the decryption_failed alert to protect against CBC attacks
- IANA registries are defined for protocol parameters.
- Premature closes no longer cause a session to be nonresumable.
- Additional informational notes were added for various new attacks on TLS

Differences between TLSv1.2 and TLSv1.1 [5]

- SHA-256 is the default digest method
- Several new cipher suites use SHA-256
- It has better ways to negotiate what signature algorithms the client supports
- Alerts are mandatory now be sent in many cases
- After a certificate_request, if no certificates are available, clients now MUST send an empty certificate list
- TLS_RSA_WITH_AES_128_CBC_SHA is now the mandatory to implement cipher suite
- Added HMAC-SHA256 cipher suites
- Removed IDEA and DES cipher suites, they are now deprecated.
- Support for the SSLv2 backward-compatible is now optional only.

[3] http://www.ietf.org/rfc/rfc4346.txt
[4] http://www.openssl.org/~bodo/tls-cbc.txt
[5] http://www.ietf.org/rfc/rfc5246.txt

Protocol Key exchange

The key exchange is used to generate a pre_master_secret known to the client and the server but not to somebody in the middle of the connection (Attacker). The pre_master_secret is then used to generate the master_secret which is used to generate the certificate "verify" and "finished" messages, encryption keys, and MAC secrets.

RSA

With RSA, key exchange and server authentication are combined. The public key may be either contained in the server's certificate or may be a temporary RSA key sent in a server key exchange message, old signatures and temporary keys cannot be replayed.

DH

DH stands for Diffie Hellman, when using DH the server supplies a certificate containing a fixed Diffie-Hellman parameter. Temporary parameters are hashed and signed to ensure that attackers cannot replay parameters. The client then verifies the certificate and signature to ensure that the parameters belong to the actual server. When using DH the client and server will generate the same pre_master_secret every time.

DHE

DHE stands for Ephemeral Diffie Hellmann, the server supplies a certificate containing **temporary Diffie-Hellman parameter signed with the servers RSA or DSS certificate**. This has the effect that it offers perfect forward secrecy. This means that even if you have compromised/broken/stolen the server private key that **you cannot decrypt past captured traffic.**

For this reason DHE and ECDHE are the recommended key exchange protocols. If for monitoring reasons decryption needs to be done we would recommend to write the Diffie Hellmann parameters to a database for every new session.

ADH

ADH stands for Anonymous Diffie Hellmann and allows completely anonymous connections, the server and client public parameters are contained in the corresponding exchange messages. Passive man-in-the-middle attacker should not be able to find the Diffie-Hellman result (i.e. the pre_master_secret), **however this method of key exchange is vulnerable to active man-in-the-middle attacks**.

ECDHE

ECDHE (or EECDH in Openssl 1.0) is DHE combined with elliptic key cryptography.

Authentication

TLS supports three authentication modes: authentication of server and client (through server and client certificate), server only authentication and anonymous connections. The algorithms available are:

No authentication

No authentication

RSA

The algorithm used to sign the certificate is RSA[6] [7]

DSS

The digital signature standard is used to sign the certificate

ECDSA

ECDSA stands for Elliptic Curve Digital Signature Algorithm; it is a variant of the Digital Signature algorithm that uses Elliptic Curve cryptography.

KRB5[8]

Kerberos credentials are used to achieve mutual authentication and to establish a master secret which is subsequently used to secure client-server communication.

PSK

Authentication takes place pre-shared keys, these symmetric keys are known to both parties prior to authenticating.

[6] http://en.wikipedia.org/wiki/RSA
[7] http://www.di-mgt.com.au/rsa_alg.html
[8] http://www.ietf.org/rfc/rfc2712.txt

Encryption

Encryption serves the purpose to transform plaintext into unreadable data through usage of an algorithm.

NULL

No encryption will take place; this is for example useful when you want to ensure the authenticity of the data

AES[9]

The Advanced Encryption Standard, previously known as Rjindael, was the winner of the NIST competition as it regarded as state of the art encryption. AES offers key sizes from 128, 192 to 256 bits of size

CAMELLIA[10]

Developed by Mitsubishi and NTT is available under a royalty free license and according to sources has been "has been evaluated favorably by several organisations, including the European Union's NESSIE project (a selected algorithm), and the Japanese CRYPTREC project (a recommended algorithm)"

RC4 / RC2

RC4 is a Stream cipher invented by Ron Rivest and was closed source until the release of the source code in 1994 to cypherpunks mailing list. There were several attacks that have been uncovered against RC4, particularly as used within WEP. RC2 is a block cipher invented by Ron Rivest in 1996 the source code was leaked to the sci.crypt UseNet group. RC2 is vulnerable to several attacks.

IDEA[11]

The International Data Encryption Algorithm is a block cipher invented by James Massey , It is still considered secure however it is patented and slower than modern ciphers. The patent will expire in 2011.

3DES

Triple-DES was created when DES was found to be vulnerable due to a key size being too small, it uses the e Data Encryption Standard cipher algorithm three times over each block.

DES

The history of DES is interesting as it was believed that the NSA tampered with the s-boxes, Wikipedia has a good summary - Simple DES is weak and should no longer be used.

[9] http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[10] http://en.wikipedia.org/wiki/Camellia_%28cipher%29
[11] http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm

Minimum industry Encryption and Key length recommendations

This summary does **not take into account the arrival of quantum computing**, large quantum computers able to crack large keys are foreseen for 2014 by the ARDA and 2018 by Prof Lloyd[12] . Shors' algorithm could then be used to break the RSA key sizes presented here below.

Recommended Asymmetric key length[13]

Period		*BSI*[14]	*NIST*[15]	*Lenstra*[16]	*FNISA*[17]
Until 2009	Minimum Recommended	1536 2048	1024	1114	1536
Until 2010	Minimum Recommended	1728 2048	1024	1152	1536
Until 2012	Minimum Recommended	1976 2048	2048	1229	2048
Until 2020	Minimum	2048	2048	1568	4096

Recommended Symmetric key length

Period		*BSI*	*NIST*	*Lenstra*	*FNISA*
Until 2009	Minimum	-	80	74	80
Until 2010	Minimum	-	80	75	80
Until 2012	Minimum	-	112	76	100
Until 2020	Minimum	-	112	82	100

Recommended Hashing algorithm and size

Period	*Type*	*BSI*	*NIST*	*Lenstra*
After 2009	-	80	148	160 minimum
After 2010	-	224	150	160 minimum
After 2012	SHA-224, SHA-256 SHA-384, SHA-512	224	152	256 minimum (SHA)
After 2020	-	224	163	256 minimum (SHA)

[12] http://synaptic-labs.com/ecosystem/context-qc-relevant-today.html
[13] http://www.rsa.com/rsalabs/node.asp?id=2264
[14] https://www.bsi.bund.de/cae/servlet/contentblob/476754/publicationFile/31104/BSI_Final_07_pdf.pdf
[15] http://csrc.nist.gov/groups/ST/toolkit/key_management.html
[16] http://people.epfl.ch/arjen.lenstra
[17] http://www.ssi.gouv.fr/site_article76.html

Client-side and Server-side Compatibility Overview

This section gives an overview over the current SSL/TLS capabilities across Operation Systems, Clients (Browsers) and Servers (Web servers). We conclude with advice on how to securely configure your SSL/TLS service and in particularly which Encryption, Authentication, Key exchange settings to use.

Throughout this document we will use the colour blue to indicate our recommended settings; this recommendation is based on compatibility and security.

Client-side: TLS / SSL Compatibility overview

In order to assess the SSL/TLS support of modern Internet browsers we had to take a look at the SSL engines they use. Some SSL stacks generally have capabilities that browsers do not make use of per default, **the lists below only reflect real default browser usage.**

- Chrome and Firefox use the NSS[18] engine
- IE5, 6, 7, 8 and Safari use Microsoft SCHANNEL[19]
- Opera and Safari (OSX) use custom SSL engines.

Default Protocol support

All browsers tested do explicitly not support SSLv2

Protocol	NSS[1]	SCHANNEL	SCHANNEL	SCHANNEL	Opera 10	Safari 4[4]
	ALL OS	XP/2K/2003[2]	7/2008R2[3]	Vista /2008[2]	All OS	OSX
SSLv2	No	No	No	No	No	No
SSLv3	Yes	Yes	Yes	Yes	Yes	Yes
TLS 1.0	Yes	Yes	Yes	Yes	Yes	Yes
TLS 1.1	No	No	Yes (disabled per default)	No	Yes	No
TLS 1.2	No	No	Yes (disabled per default)	No	Yes	No

Default Key exchange support

We recommend using Ephemeral Diffie Hellmann paired with either RSA or DSS as signature.

Algorithm	NSS[1]	SCHANNEL	SCHANNEL	SCHANNEL	Opera 10	Safari 4[4]
	ALL OS	XP/2K/2003[2]	7/2008R2[3]	Vista /2008[2]	All OS	OSX
RSA	Yes	Yes	Yes	Yes	Yes	Yes
DHE-RSA	Yes	No	No	No	Yes	Yes
DHE-DSS	Yes	Yes	Yes	Yes	Yes	Yes
ECDHE-RSA	Yes	No	Yes	Yes	No	No
ECDH-RSA	Yes	No	No	No	No	No
ECDHE-ECDSA	Yes	No	Yes	Yes	No	No
ECDH-ECDSA	Yes	No	No	No	No	No
ADH	No	No	No	No	No	No

1 Firefox, Google chrome (New) – All OS | 2 IE 7 & IE 8 & Safari | 3 IE8 & IE9 (**not** Safari – see VISTA column for Safari 7/2008R2 support)| 4 OSX

Recommended

[18] http://www.mozilla.org/projects/security/pki/nss/
[19] http://msdn.microsoft.com/en-us/library/windows/desktop/ms678421(v=vs.85).aspx

RSA support

RSA public-key cryptosystem is an asymmetric encryption method; it can be used for signatures as well as encryption. In SSL/TLS RSA is used during key exchange (handshake). RSA bases its security on the length of the modulus that must be factored. The bigger the modulus the harder it is to break the algorithm.

Browser supported RSA key size, DH and SRP [20]

These are the key sizes that are supported by major Browsers, there is no client side restriction to use 1024 bit instead of 2048, and additionally 1024 bit are considered weak by today's standards.

RSA Modulus	NSS[1]	SCHANNEL	SCHANNEL	SCHANNEL	Opera 10	Safari 4[4]
	ALL OS	XP/2K/2003 [2]	7/2008R2 [3]	Vista /2008 [2]	ALL OS	OSX
1024	Yes	Yes	Yes	Yes	Yes	Yes
2048	Yes	Yes	Yes	Yes	Yes	Yes
4096	Yes	Yes	Yes	Yes	Yes	Yes
Note:					Generally no limit; 4k limit on client cert	

Default supported Ciphers [21]

In order for this list to stay focused on best practices we list modern or strong ciphers only.

Cipher	Size	NSS[1]	SCHANNEL	SCHANNEL	SCHANNEL	Opera 10	Safari 4[4]
		ALL OS	XP/2K/2003 [2]	7/2008R2 [3]	Vista /2008 [2]	ALL OS	OSX
AES	128	Yes	No [19]	Yes	Yes	Yes	Yes
AES	256	Yes	No [19]	Yes	Yes	Yes	Yes
AES-GCM	256	No	No	Yes	No	No	No
RC4	128	Yes	Yes	Yes	Yes	Yes	Yes
Camellia	128	Yes	No	No	No	No	No
Camellia	256	Yes	No	No	No	No	No
3DES	168	Yes	Yes	Yes	Yes	Yes	Yes

1 Firefox, Google chrome (New) – All OS | 2 IE 7 & IE 8 & Safari | 3 IE8 & IE9 (**not** Safari – see VISTA column for Safari 7/2008R2 support)| 4 OSX

Recommended

[20] http://msdn.microsoft.com/en-us/library/bb931357%28VS.85%29.aspx
[21] With heavy support from SSLLAB (Ivan Ristic)

Default ECC support

Elliptic curve cryptography bases on a discrete logarithm problem, ECC needs less key size to achieve the same strength then RSA, as an example, an ECC 160-bit field offers the same resistance as an 1024-bit RSA modulus. This allows for smaller keys and offers improved performance. Unfortunately ECC is not widely supported in Browser as of yet, but certainly will be in the future. We are currently not aware of any Certificate authority that allows you to buy ECC certificates.

Elliptic key cryptography

Curve size	NSS [1]	SCHANNEL	SCHANNEL	SCHANNEL	Opera 10	Safari 4[4]
	All OS	XP/2K/2003[2]	7[3]/2008R2	Vista[2]/2008	ALL OS	OSX
P-256	Yes	No	Yes	Yes	No	No
P-348	Yes	No	Yes	Yes	No	No
P-521	Yes	No	**No**	Yes	No	No

1 Firefox, Google chrome (New) – All OS | 2 IE 7 & IE 8 & Safari | 3 IE8 & IE9 (**not** Safari – see VISTA column for Safari 7/2008R2 support)| 4 OSX

According to Microsoft support for P521 mode has been removed from Windows 7 and 2008R2 due to not being part of the official NIST Suite B.

Recommended

Server-Side: TLS / SSL Compatibility overview

Default protocol support

This matrix shows the protocol support of modern web servers - There is no reason to continue supporting SSLv2.

Protocol	IIS6 [1]	IIS7 [2]	IIS7.5 [3]	mod_ssl	mod_gnutls	JSSE [4]	NSS [5]
SSLv2	Yes	Yes	Yes	Yes	No	Yes	Yes
SSLv3	Yes	Yes	Yes	Yes		Yes	Yes
TLS 1.0	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TLS 1.1	No	Yes	Yes (disabled per default)	No	Yes	No	Yes
TLS 1.2	No	No	Yes (disabled per default)	No	Yes (disabled per default)	No	Yes

* See appendix on how to enable TLS 1.2 support on IIS 7.5

Default key exchange support

We recommend offering ephemeral Diffie Hellmann paired with either RSA or DSS as signature

Algorithm	IIS6 [1]	IIS7 [2]	IIS7.5 [3]	mod_ssl	mod_gnutls	JSSE [422]	NSS [5]
RSA	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DHE-RSA	No	Yes	Yes	Yes	Yes	Yes	Yes
DHE-DSS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ECDHE-RSA	No	Yes	Yes	Yes[23] [24]	No	No	No (Default)
ECDH-RSA	No	No	No	Yes	No	No	No (Default)
ECDHE-ECDSA	No	Yes	Yes	Yes	No	No	No (Default)
ECDH-ECDSA	No	No	No	Yes	No	No	No (Default)
ADH		No	No	No	No	No	No

1 Windows 2003 | 2 Windows 2008 | 3 Windows 2008 R2 | 4 Tomcat | 5 Network Security Services (Apache, Redhat, Sun Java Enterprise..)

Recommended

[22] http://download.oracle.com/javase/6/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider
[23] https://issues.apache.org/bugzilla/show_bug.cgi?id=40132
[24] ECCdraft suite – after 1.0 included in ALL

Default RSA size support

RSA public-key cryptosystem is an asymmetric encryption method (public-key cryptography), it can be used for signing as well as encryption. In SSL/TLS RSA is used during key exchange (handshake). RSA bases its security on the length of the modulus that must be factored. The bigger the modulus the harder it is to break the algorithm.

Server RSA key size, DH and SRP prime support

This list the key sizes that are supported by Major Web servers, there is no server side restriction to use 1024 bit instead of 2048. Performance issues should not be of concern for most providers; TLS introduced caching and session resumption, reducing the RSA computations to a minimum. On windows the tool "Harden SSL/TLS" also allows tweaking the TLS session caching for IIS.

RSA Modulus	IIS6 [1]	IIS7 [2]	IIS7.5 [3]	mod_ssl	mod_tls [4]	JSSE	NSS [5]
1024	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2048	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4096	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Server Cipher support [i]

In order for this list to stay focused on best practices we display modern or strong ciphers only and beta versions of SSL engines are taken into account.

Cipher	Size	IIS6 [1]	IIS7 [2]	IIS7.5 [3]	mod_ssl	mod_tls [4]	JSSE	NSS [5]
AES	128	No	Yes	Yes	Yes	Yes	Yes	Yes
AES	256	No	Yes	Yes	Yes	Yes	Yes	Yes
AES-GCM	128	No	No	Yes	Yes	No	No	No
AES-GCM	256	No	No	Yes	Yes	No	No	No
RC4	128	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Camellia	128	No	No	No	Yes	Yes	No	Yes
Camellia	256	No	No	No	Yes	Yes	No	Yes
3DES	156	Yes	Yes	Yes	Yes	Yes	Yes	Yes

1 Windows 2003 | 2 Windows 2008 | 3 Windows 2008 R2 | 4 Tomcat | 5 Network Security Services (Apache, Redhat, Sun Java Enterprise...)

Recommend Server-Side SSL configuration - Putting it all together -

Taking into account the previous client and server compatibility matrixes it is apparent that the best setup to use has changed over the years. Protocols have been enhanced and weaknesses patched and encryption strengthened.

IIS7.5

These are the cipher suites that offer most security and compatibility, no SSLv2 and SSlv3 support should be provided at all.

Cipher suite name	Protocol	KeyX	Auth	Enc	bit	Hash	Comp.
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384	TLS 1.2	ECDHE	ECDSA	AES	256	SHA2	
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA*	TLS 1.0	ECDHE	RSA	AES	256	SHA	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA*	TLS 1.0	ECDHE	RSA	AES	128	SHA	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	DHE	RSA	AES	256	SHA	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	DHE	RSA	AES	128	SHA	
TLS_RSA_WITH_RC4_128_SHA	TLS 1.0	RSA	RSA	RC4	128	SHA	
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS 1.0	DHE	DSS	3DES	168	SHA	

- Firefox & Chrome (NSS)
- Opera
- Windows XP/2000/2003 (IE7/IE8, Safari)
- Windows 7/2008R2 (IE8) (Safari excluded)
- Windows Vista/2008R1 (IE8/IE7 ,Safari)
- Safari (MacOSx)

* RSA chosen over ECDSA due to the current lack of ECC certificate authorities, once ECC certificates are available we recommend offering TLS_ECDHE_**ECDSA**_WITH_AES_256_CBC_SHA

IIS7

These are the cipher suites that offer most security and compatibility for IIS7

Cipher suite name	Protocol	KeyX	Auth	Enc	bit	Hash	Comp.
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA*	TLS 1.0	ECDHE	RSA	AES	256	SHA	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA*	TLS 1.0	ECDHE	RSA	AES	128	SHA	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	DHE	RSA	AES	256	SHA	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	DHE	RSA	AES	128	SHA	
TLS_RSA_WITH_RC4_128_SHA	TLS 1.0	RSA	RSA	RC4	128	SHA	
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS 1.0	DHE	DSS	3DES	168	SHA	

- Firefox & Chrome
- Opera
- Windows XP/2000/2003 (IE7/IE8) + Safari (All windows OS up to 2008R2)
- Windows 7/2008R2 (IE8)
- Windows Vista/2008R1 (IE8/7)
- Safari (MacOSx)

* Chosen over ECDSA due to the current lack of ECC certificate authorities, once ECC certificates are available we recommend offering TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

IIS6 [25] [26]

These are the cipher suites that offer most security and compatibility for IIS6

Cipher suite name	Protocol	KeyX	Auth	Enc	bit	Hash	Comp.
TLS_DHE_RSA_WITH_AES_256_CBC_SHA*	TLS 1.0	DHE	RSA	AES	256	SHA	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA*	TLS 1.0	DHE	RSA	AES	128	SHA	
TLS_RSA_WITH_RC4_128_SHA	TLS 1.0	RSA	RSA	RC4	128	SHA	
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS 1.0	DHE	DSS	3DES	168	SHA	

* IIS6 will support AES only after the installation of a Hotfix (which is recommended)

- Firefox & Chrome
- Opera
- Windows XP/2000/2003 (IE7/IE8) for Chrome + Safari (All windows OS up to 2008R2)
- Windows 7/2008R2 (IE8)
- Windows Vista/2008R1 (IE8/7)
- Safari (MacOSx)

[25] http://support.microsoft.com/?scid=kb;en-us;245030&x=14&y=11
[26] http://www.gorlani.com/publicprj/CipherControl/

Apache https / Tomcat (OpenSSL 1.0)

We are aware that OpenSSL 1.0 is currently beta only, this guide however was intended to be future proof [27] to a certain degree, to achieve this Elliptic Cryptography is mandatory.

Cipher suite name	Protocol	KeyX	Auth	Enc	bit	Hash	Comp.
ECDHE-RSA-AES256-SHA*	TLS 1.0	ECDHE	ECDSA	AES	256	SHA	
ECDHE-RSA-AES128-SHA*	TLS 1.0	ECDHE	ECDSA	AES	128	SHA	
DHE-RSA-AES256-SHA	TLS 1.0	DHE	RSA	AES	256	SHA	
DHE-RSA-AES128-SHA	TLS 1.0	DHE	RSA	AES	128	SHA	
TLS_RSA_WITH_RC4_128_SHA	TLS 1.0	RSA	RSA	RC4	128	SHA	
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	TLS 1.0	DHE	DSS	3DES	168	SHA	

- Firefox & Chrome
- Opera
- Windows XP/2000/2003 (IE7/IE8) - Chrome + Safari (All windows OS up to 2008R2)
- Windows 7/2008R2 (IE8)
- Windows Vista/2008R1 (IE8/7)
- Safari (MacOSx)

* Chosen over ECDSA due to the current lack of ECC certificate authorities, once ECC certificates are available we recommend offering TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

[27] http://mail-archives.apache.org/mod_mbox/httpd-cvs/200911.mbox/%3C20091110075514.166A6238890A@eris.apache.org%3E

Server configurations – undocumented behaviour

This section covers configuration issues with regards to enabling particular cipher suites, it is not meant to serve as a general documentation but lists the results of our research as well as undocumented features.

```
.text:6C2FBFB1
.text:6C2FBFB1 loc_6C2FBFB1:                           ; CODE XREF: SslReadRegOptions(int)+2B0↑j
.text:6C2FBFB1                                         ; SslReadRegOptions(int)+4E37↓j
.text:6C2FBFB1                 lea     eax, [ebp+phkResult]
.text:6C2FBFB4                 push    eax             ; phkResult
.text:6C2FBFB5                 push    edi             ; samDesired
.text:6C2FBFB6                 push    ebx             ; ulOptions
.text:6C2FBFB7                 push    offset aCiphers ; "Ciphers"
.text:6C2FBFBC                 push    dword_6C3232C8  ; hKey
.text:6C2FBFC2                 call    __imp__RegOpenKeyExW@20 ; RegOpenKeyExW(x,x,x,x,x)
.text:6C2FBFC8                 test    eax, eax
.text:6C2FBFCA                 jnz     loc_6C300B28
.text:6C2FBFD0
.text:6C2FBFD0 loc_6C2FBFD0:                           ; CODE XREF: SslReadRegOptions(int)+4E3F↓j
.text:6C2FBFD0                 mov     [ebp+var_18], ebx
.text:6C2FBFD3                 cmp     ?g_cAvailableCiphers@@3KA, ebx ; ulong g_cAvailableCiphers
.text:6C2FBFD9                 jbe     short loc_6C2FC02D
.text:6C2FBFDB                 mov     esi, offset ?g_AvailableCiphers@@3PAUcsel@@A ; csel * g_AvailableCiphers
.text:6C2FBFE0
.text:6C2FBFE0 loc_6C2FBFE0:                           ; CODE XREF: SslReadRegOptions(int)+33F↓j
.text:6C2FBFE0                 mov     eax, [esi]
.text:6C2FBFE2                 mov     [ebp+var_24], eax
```

Figure 1 - IDA Pro Free / Disassembly of schannel.dll

General Note

ECC Certificates cannot be purchased yet, apparently due to a license problem with Certicom [28] [29]. The Root ECC certificates themselves already ship with various browsers[30] [31]

IIS 7.5 / Windows 7 / Windows 2008R2

TLS 1.2 support can be enabled in IIS 7.5 by setting the particular registry key or by using Harden-SSL/TLS.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\P
rotocols

Key : SSL 2.0\Server
DWORD ENABLED = 0
Key : SSL 3.0\Server
DWORD ENABLED = 0
Key : TLS 1.0\Server
DWORD ENABLED = 1
Key : TLS 1.1\Server
DWORD ENABLED = 1
Key : TLS 1.2\Server
DWORD ENABLED = 1

Comments: If no DWORD is present the default applies. Default = TLS 1.0, SSLv3
enabled. TLS 1.2 disabled.
```

[28] http://serverfault.com/questions/91212/do-any-well-known-cas-issue-elliptic-curve-certificates
[29] http://www.certicom.com/pdfs/FAQ-TheNSAECCLicenseAgreement.pdf
[30] http://code.google.com/p/chromium/issues/detail?id=4385
[31] https://investor.verisign.com/releasedetail.cfm?ReleaseID=360952

- Default order and an exact list of ciphers can either be set as a group policy or by using Harden-SSL
- P521 mode can be re-enabled by manually adding P521 to the Group Cipher list

IIS 6 / Windows 2003

- AES 128 and AES 256 cipher support can be added by using Hotfix 192447

Apache httpd / Tomcat (OpenSSL)

- Enable:ALL includes EC ciphers since Openssl 1.0, ECCDRAFT had to be enabled previously

General Recommendations

Minimum SSL configuration

- Use a private key that is **at least 2048** bits long
  (See section "Minimal symmetric Key length")
- Do not offer ciphers below 128 bit
  (See section "Minimal asymmetric Key length")
- Do not support SSLv2
  (see section "SSLv2 Technical details")
- Do not offer Anonymous Diffie Hellman support (ADH)
- Do not reuse keys across certificates and generate new keys for every certificate you request
- Do offer TLS 1.0 and/or better support

Recommended SSL configuration

- Offer Elliptic key cryptography as preferred cipher
- Offer AES as encryption algorithm
- Offer a minimum encryption key length 128-bit
- Offer key exchange that that offer perfect forward secrecy (DHE)
- Offer an RSA key size needs to be **at least 2048** bits strong
- Drop support for SSLv2 and SSLv3 (See Browser compatibility chart)
- Restrict protocol support TLS 1.0 or better support (See Browser compatibility chart)
- Use Client certificates as an additional layer to authenticate clients

Sources

1. http://www.ssllabs.com
2. https://www.ssllabs.com/projects/rating-guide/index.html
3. http://www.foundstone.com/us/resources/proddesc/ssldigger.htm
4. https://www.mikestoolbox.net/
5. http://extendedsubset.com/
6. http://www.wikipedia.org

Thanks

We would like to thank Ivan Ristic (SSL Labs) and Marsh Ray for the support and the information provided. We would like to thank Opera for their feedback on Opera TLS compatibility.

Disclaimer

The Information is believed to be accurate by the time of writing.

Copyright

Appendix

Example code - Listing ciphers (Windows7 & Windows 2008R2) [32]

Windows 7 and Windows 2008 allow for a new programmatic way to list and set cipher suites.

```
#include <stdio.h>
#include <windows.h>
#include <bcrypt.h>


void main()
{

   HRESULT Status = ERROR_SUCCESS;
   DWORD   cbBuffer = 0;
   PCRYPT_CONTEXT_FUNCTIONS pBuffer = NULL;

    Status = BCryptEnumContextFunctions(
        CRYPT_LOCAL,
        L"SSL",
        NCRYPT_SCHANNEL_INTERFACE,
        &cbBuffer,
        &pBuffer);
    if(FAILED(Status))
    {
        printf_s("\n**** Error 0x%x returned by BCryptEnumContextFunctions\n", Status);
        goto Cleanup;
    }

    if(pBuffer == NULL)
    {
        printf_s("\n**** Error pBuffer returned from BCryptEnumContextFunctions is
null");
        goto Cleanup;
    }

    printf_s("\n\n Listing Cipher Suites ");
    for(UINT index = 0; index < pBuffer->cFunctions; ++index)
    {
        printf_s("\n%S", pBuffer->rgpszFunctions[index]);
    }

Cleanup:
    if (pBuffer != NULL)
    {
        BCryptFreeBuffer(pBuffer);
    }
}
```

[32] http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp892.pdf

Example Code - Setting preferred cipher (Windows7 & Windows 2008R2)

```
#include <stdio.h>
#include <windows.h>
#include <bcrypt.h>


void main()
{

    SECURITY_STATUS Status = ERROR_SUCCESS;
    LPWSTR wszCipher = (L"RSA_EXPORT1024_DES_CBC_SHA");

    Status = BCryptAddContextFunction(
                CRYPT_LOCAL,
                L"SSL",
                NCRYPT_SCHANNEL_INTERFACE,
                wszCipher,
                CRYPT_PRIORITY_TOP);
}
```

Code - Remove ciphers 33

```
#include <stdio.h>
#include <windows.h>
#include <bcrypt.h>


void main()
{

    SECURITY_STATUS Status = ERROR_SUCCESS;
      LPWSTR wszCipher = (L"TLS_RSA_WITH_RC4_128_SHA");

    Status = BCryptRemoveContextFunction(
                CRYPT_LOCAL,
                L"SSL",
                NCRYPT_SCHANNEL_INTERFACE,
                wszCipher);
}
```

[33] http://msdn.microsoft.com/en-us/library/bb870930(VS.85).aspx

Default Windows SCHANNEL cipher support

The windows Schannel interface is used by Internet Explorer, Chrome, Safari and other third party applications. It represents the easiest way to implement TLS/SSL under windows.

Note how Vista and Server 2008 R1 share the same cipher list but Windows 7 lacks support for P512, but introduces TLS 1.2 support (with SHA2 and AES GCM).

Note that the applications using the SCHANNEL interface specify which protocol version and which cipher suites they want to support. For instance IE7&8 have SSLv2 disabled by default and will not use NULL ciphers and Chrome as well as Safari currently do not use TLS1.2 ciphers and features even if provided by Windows 7.

Windows 7 and Windows Server 2008R2

```
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P384
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_MD5
SSL_CK_RC4_128_WITH_MD5
SSL_CK_DES_192_EDE3_CBC_WITH_MD5
TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_NULL_SHA
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
```

Windows Vista AND Windows Server 2008 R1

TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P521
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P521
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P521
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P521
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_MD5
SSL_CK_RC4_128_WITH_MD5
SSL_CK_DES_192_EDE3_CBC_WITH_MD5
TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA

Windows XP,2000,2003

TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA

Default Browser support

This section covers the TLS/SSL support offered by Internet browsers; the first cipher in the list is the preferred cipher by that particular browser. This list is different than the SCHANNEL list as every application can request a special subset of ciphers and protocols.

IE6, 7, 8 – Windows XP, 2003, 2000

```
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
```

IE7, IE 8 – Windows Vista

```
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_RC4_128_MD5
```

Firefox, Google Chrome (NSS) - All Operation Systems

```
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88)
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA (0x87)
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39)
TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x38)
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA (0xc00f)
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA (0xc005)
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x84)
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (0xc007)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x45)
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA (0x44)
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)
TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x32)
TLS_ECDH_RSA_WITH_RC4_128_SHA (0xc00c)
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xc00e)
TLS_ECDH_ECDSA_WITH_RC4_128_SHA (0xc002)
```

```
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xc004)
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x41)
TLS_RSA_WITH_RC4_128_MD5 (0x04)
TLS_RSA_WITH_RC4_128_SHA (0x05)
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008)
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16)
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x13)
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA (0xc00d)
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc003)
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (0xfeff)
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x0a)
```

Opera

```
0039  TLS_DHE_RSA_WITH_AES_256_CBC_SHA
0038  TLS_DHE_DSS_WITH_AES_256_CBC_SHA
0037  TLS_DH_RSA_WITH_AES_256_CBC_SHA
0036  TLS_DH_DSS_WITH_AES_256_CBC_SHA
0035  TLS_RSA_WITH_AES_256_CBC_SHA
0033  TLS_DHE_RSA_WITH_AES_128_CBC_SHA
0032  TLS_DHE_DSS_WITH_AES_128_CBC_SHA
0031  TLS_DH_RSA_WITH_AES_128_CBC_SHA
0030  TLS_DH_DSS_WITH_AES_128_CBC_SHA
002F  TLS_RSA_WITH_AES_128_CBC_SHA
0005  TLS_RSA_WITH_RC4_128_SHA
0004  TLS_RSA_WITH_RC4_128_MD5
0013  TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
000D  TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
0016  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
0010  TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
000A  TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

TLS/SSL Interop Test services

1. GNUTLS - http://www.gnu.org/software/gnutls/server.html (Gnutls)
2. Certicom – ECC Interop (recommended)
3. Mikes toolbox
4. Microsoft IIS 7.5 interop (Schannel)
5. Fedora ECC Test server (NSS)
6. Sun's ECC/TLS test server
7. Sun's JES Web Server 7.0 ECC/TLS test server

With heavy support from SSLLAB (Ivan Ristic)