CENTER FOR CYBERSIKKERHED

Vejledning

Sikker brug af Transport Layer Security

Retningslinjer for sikker brug af Transport Layer Security (TLS).

Indhold

CENTER FOR CYBERSIKKERHED

Kastellet 30
2100 København Ø
Telefon: + 45 3332 5580
E-mail: cfcs@cfcs.dk

1. udgave oktober 2020.

Indledning

Nærværende retningslinjer henvender sig primært til personer, der skal fastlægge sikkerhedskrav til brugen af Transport Layer Security (TLS). Dette kan eksempelvis være i forbindelse med anskaffelse af TLS-baserede systemer og -tjenester, alternativt ved konfiguration eller revision af sådanne systemer og tjenester.

TLS og dens forgænger, Secure Sockets Layer (SSL), er protokoller der er designet til at sikre kommunikationen over et netværk gennem brug af krypteringsteknologi. Protokollerne anvendes i udbredt grad i applikationer til web-browsing, e-mail, instant messaging, voice over IP (VoIP) m.v. Eksempelvis kan web-applikationer anvende TLS til at sikre, at kommunikation mellem web-server og web-browseren krypteres.

Valget af konfigurationsværdier for de enkelte TLS-parametre afhænger i høj grad af den konkrete TLS-implementering, og derfor kan der ikke gives en entydig anbefaling på hvorledes TLS som helhed skal konfigureres.

I denne vejledning har vi derfor valgt at kategorisere mulige konfigurationsværdier for de enkelte sikkerhedsparametre, så det fremgår om værdien anses for at være ”God”, ”Tilstrækkelig”, ”Bør udfases” eller ”Utilstrækkelig”. Kategoriseringen fremgår af bilaget.

Generelt bør der kun vælges konfigurationsværdier for sikkerhedsparametrene, der er kategoriseret som ”God” eller ”Tilstrækkelig”. Værdier fra kategorien ”Bør udfases” frarådes. Disse kan dog anvendes i en overgangsperiode, hvis der samtidig fastsættes en dato for, hvornår der skal ske en udfasning. Værdier, der kategoriseres som ”Utilstrækkelig” bør ikke anvendes.

Som led i den nationale cyber- og informationssikkerhedsstrategi blev det i 2019 besluttet, at de statslige myndigheder skal efterleve en række tekniske minimumskrav til it-sikkerheden med henblik på at sikre et højt fælles it-sikkerhedsniveau i staten. Statslige myndigheder skal derfor som minimum anvende TLS version 1.2.

Bilag ”Oversigt over konfigurationsparametre” giver den fulde liste over alle sikkerhedsrelevante TLS-parametre med relaterede og anbefalede konfigurationsværdier.

Introduktion til TLS

Behovet for at sikre data og informationer i transit over åbne netværk har i mange år været drøftet i forbindelse med behandlingen af personoplysninger. Også inden for andre områder er behovet for beskyttelse af data og informationers integritet og fortrolighed relevant. Dette gælder for eksempel online handel og medarbejderes eksterne adgang til deres arbejdsplads' informationssystemer.

I 1994 introducerede den amerikanske browserproducent Netscape "Secure Socket Layer" (SSL) 1.0 med henblik på at kunne tilbyde en sikker kanal mellem brugerens browser og en web-server. Specifikationerne for SSL 1.0 blev aldrig frigivet og protokollen blev kritiseret for at anvende dårlige (svage) krypto-algoritmer. Senere samme år frigav Netscape specifikationerne for SSL 2.0, som indeholdt en række forbedringer i forhold til version 1.0, men igen viste det sig, at der var en række problemer i implementeringen. Derfor udviklede Microsoft i 1995 deres egen variant – "Private Communication Technology" (PCT). I 1998 kom så SSL 3.0 og i 1999 blev der taget initiativ til at forene SSL med Microsofts PCT. Dette førte til TLS 1.0. Aktuelt har flere browser-leverandører udmeldt, at deres browsere ikke længere understøtter SSL og de tidlige versioner af TSL (version 1.0 og 1.1).

Efterfølgende blev der i 2005, 2008 og 2016 frigivet specifikationer for TLS 1.1, 1.2 og 1.3. Sidstnævnte er den seneste og aktuelle version af TLS. Udviklingen har blandt andet betydet, at sikkerheden er blevet øget, men der er også sket en forenkling af den indledende kommunikation mellem klient og server.

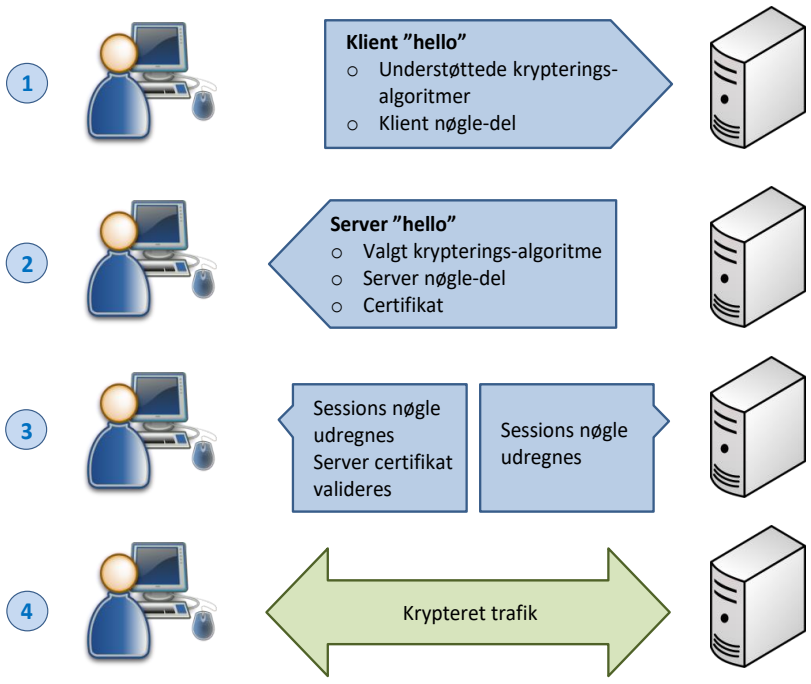
Desværre ses der stadig rigtig mange implementeringer baseret på såvel SSL som tidlige versioner af TLS. Dette betyder reelt, at sådanne implementeringer er sårbare på en række områder.

Hvordan virker TLS

En forbindelse mellem en klient og en server, der er sikret med TLS kaldes en TLS-session. TLS-sessionen kan opdeles i henholdsvis en etableringsfase (”handshake”) og en applikationsfase. I etableringsfasen bliver en række sessionsparametre udvekslet/fastlagt, herunder[1]:

- Hvilken SSL/TLS version, der skal anvendes
- Hvilke kryptografiske algoritmer, der skal anvendes
- Certifikat som serveren anvender til at autentificere sig over for klienten
- Om klienten skal præsentere et certifikat over for serveren – og hvis ja hvilket
- Hvilken ciffersuite (cipher suite)[2] skal anvendes til udvekslingen af applikationsdata.

Det er klienten, der initierer etableringsfasen med et kald til serveren, og når denne fase er gennemført fortsættes til applikationsfasen hvor TSL-sessionen er til rådighed som en sikker tunnel mellem klient og server. Figur 1 illustrerer etableringsfasen mellem klient og server ved anvendelse af TLS 1.3.



Figur 1 Illustration af etableringsfasen for TLS v1.3

[1] Hvilke sessions-parametre der udveksles afhængig af SSL/TLS version.
[2] I TLS 1.3 specifikationerne ændrede man den tidligere fortolkning af hvad termen ”cipher suite” dækkede over. Tidligere dækkede termen over krypteringsalgoritmerne for nøgleudveksling, certifikatverifikation, bulk kryptering og hashing. I TLS 1.3 dækker termen nu kun over krypteringsalgoritmer for bulk kryptering og hashing.

TLS er understøttet af en lang række forskellige software-biblioteker, som udviklere kan basere deres applikationer på. Dette er en klar fordel, da det forenkler udviklingen af applikationer og sikrer, at den anvendte TLS-kode er velafprøvet.

Anbefaling
Ved valg af TLS-software-bibliotek anbefales følgende:

- Anvend altid software-biblioteker fra anerkendte kilder
- Anvend altid seneste version af det TLS-software-bibliotek, der er valgt
- Anvend kun de indstillinger, der er nødvendige for at imødekomme forretningens krav
- Følg i øvrigt de retningslinjer som følger med TLS-software-biblioteket

Derudover skal man være opmærksom på særligt to relevante problemstillinger omkring anvendelsen af krypteringsfunktioner, som beskrives i efterfølgende afsnit "generelle opmærksomhedspunkter"

Generelle opmærksomhedspunkter

Brug af soft- eller hardware baserede moduler til generering af tilfældige tal

Stort set alle krypteringsalgoritmer har behov for adgang til en funktion, der kan generere tilfældige tal. Hardware-baserede moduler til generering af tilfældige tal, anses normalt for at være i stand til at generere tilfældige tal langt hurtigere og med en bedre kvalitet end softwarebaserede moduler, der "kun" kan generere det, som mange betegner som pseudo-tilfældige tal. Graden af tilfældighed i de enkelte tal er afgørende for den samlede sikkerhed i forbindelse med kryptering. Hardware-baserede enheder til generering af tilfældige tal er dog som udgangspunkt væsentlig dyrere end de software-baserede løsninger. Derfor ses de stort set kun anvendt der, hvor kravet til sikkerhed er ekstra højt, eksempelvis inden for forsvaret og i den finansielle sektor.

Kontrol over klienter

I forbindelse med valg af værdier for de enkelte konfigurationsparametre bør man overveje i hvilken grad man har kontrol over potentielle klienter. Dette er vigtigt, da enhver TLS-session kun kan etableres på et sikkerhedsmæssigt niveau, der understøttes af server såvel som den enkelte klient.

Kommunikation internt i en organisation: Når en løsning anvendes internt i en organisation vil man i højere grad være i stand til at styre konfigurationen af de klienter, der kan/må tilgå en løsning, og man vil derfor have en høj grad af kontrol. Det anbefales derfor, at man fravælger enhver brug af SSL samt TLS i de tidlige versioner (før TLS 1.2).

Kommunikation med eksterne – fx med en borger via internettet: I denne situation vil man have en mindre grad af styring med konfigurationen af klienterne. Valget af værdien af de enkelte konfigurationsparametre bør hvile på en afvejning af behovet for kompatibilitet kontra behovet for sikkerhed. Det endelige valg bør altid træffes på baggrund af en risikovurdering. Det anbefales dog, at man aktivt fravælger klienter, der kun understøtter SSL (dvs. SSL 1.0, SSL 2.0 og SSL 3.0).

Bemærk, at der for statslige myndigheder, som tidligere nævnt, er et teknisk minimumskrav, der dikterer en anvendelse af minimum TLS version 1.2 for almindelige web-tjenester.

Gennemgang af relevante sikkerhedsparametre for TLS

I dette afsnit gennemgås de relevante TLS sikkerhedsparametres anvendelse. Man skal være opmærksom på, at en konkret implementering af TLS kan have begrænsninger, der ikke muliggør understøttelse af alle anbefalinger i denne publikation. Dette kan i yderste konsekvens betyde, at man ud fra et it-sikkerhedsmæssigt perspektiv bør finde en alternativ implementering af TLS for at opnå det anbefalede sikringsniveau.

TLS versioner
TLS-versioner angiver, som navnet antyder, hvilken version af TLS-protokollen, der skal anvendes i kommunikationen mellem klient og server.

Anbefaling
Center for Cybersikkerhed anbefaler, at man som minimum anvender TLS version 1.2. Tidligere TLS-versioner bør udfases og alle SSL-versioner helt undgås, da protokollerne indeholder kendte sårbarheder, som ikke kan fjernes.

Konfigurationsparametrer for krypteringsalgoritmer
Sikkerheden i en TLS-forbindelse afhænger i høj grad af hvilke algoritmer, der anvendes i de forskellige faser. Der skal fastlægges algoritmer for:

- Nøgleudveksling
- Certifikatverifikation
- Bulk kryptering (dvs. kryptering i applikationsfasen)
- Hashing

Anbefaling
Det anbefales, at valget af værdier for konfigurationsparametrene knyttet til understøttede algoritmer for såvel nøgleudveksling, certifikatverifikation, bulk kryptering og hashing alle sker fra kategorierne ”God” eller ”Tilstrækkelig”.

Baseret på en risikovurdering kan valg fra kategorien ”Bør udfases” ske i en afgrænset overgangsperiode såfremt, der er forretningsmæssige behov herfor. Dette kan eksempelvis være, hvis man vil sikre kompatibiliteten med ældre it-systemer under udfasning. Der bør i så fald være igangsat aktiviteter, der på sigt muliggør en opgradering til et mere sikkert valg.

Nøgleudveksling
I etableringsfasen skal server og klient ”enes” om, hvordan krypteringsnøglen til brug i applikationsfasen, fastlægges. Til dette findes der en lang række forskellige algoritmer som for eksempel ECDHE (Elliptic Curve Diffie-Hellman), DHE (Diffie-Hellman baseret på finite-fields) samt RSA (Rivest–Shamir–Adleman).

Det matematiske grundlag for disse algoritmer ligger uden for denne publikations rammer, men der er en grundlæggende forskel på hvilke oplysninger, der udveksles i etableringsfasen, samt hvor god sikkerheden er omkring de i applikationsfasen transmitterede informationer. I tilfælde af, at serverens private nøgle på et tidspunkt kompromitteres, vil en angriber med adgang til den private nøgle have mulighed for at dekryptere transmitterede data, hvis disse er eller bliver opsnappet. Anvendelsen af ECDHE/DHE understøtter "Forward secrecy", der betyder at klient og server ikke direkte anvender deres egne nøgler til kryptering i applikations-fasen. I stedet "aftales" det, at der anvendes en temporær nøgle i hver session, som destrueres efter brug. Hvis en temporær nøgle bliver kompromitteret vil det betyde, at kun informationer hørende til samme session kan dekrypteres.

Anbefaling

Det anbefales, at man kun understøtter nøgleudveksling baseret på ECDHE, og kun en eller flere af følgende kurver: secp384r1, secp256r1, x448 og x25519.
For øvrige elliptiske kurver, finit-fields grupper og algoritmer henvises til bilag.

Certifikat-verifikation

I TLS autentificeres serveren over for klienter ved brug af et X.509[3] certifikat. Klienter kan på samme måde afkræves at autentificere sig over for serveren. Et certifikat kan være udstedt og underskrevet af indehaveren selv ("self-signed" certifikat) eller af en ekstern CA[4]. Hvis muligt bør man vælge at benytte sig af en ekstern CA, der er kendt af klienterne. DANE[5] kan anvendes til at angive overfor klienter, hvilken ekstern CA, der er betroet at udstede certifikater for domænet eller hvilket "self-signed" certifikat, der er gyldigt.

For at klienten kan verificere serverens certifikat, skal klienten understøtte den signeringsalgoritme, der er anvendt af CA'en i forbindelse med udstedelsen af server-certifikatet. I dag er der kun to algoritmer som anses for gode i den sammenhæng: ECDSA og RSA.

Anbefaling

Bemærk, at der ved anvendelse af RSA anbefales en minimums nøglelængde på 2048 bit.

Bulk kryptering

Når etableringsfasen er overstået, overgår TLS-sessionen som nævnt til applikations-fasen hvor alt data krypteres mellem klient og server. Den krypteringsalgoritme, der skal anvendes i den sammenhæng skal ligeledes aftales i etableringsfasen.

Anbefaling

Det anbefales, at man understøtter en eller flere af følgende tre algoritmer: AES-256-GCM, ChaCha20-Poly1305 og AES-128-GCM. For øvrige mulige algoritmer henvises til bilag.

[3] X.509 er en standard, der definerer formatet af et certifikat for den offentlige nøgle i en PKI.
[4] CA (Certificate Authority) er en funktion hvis opgave er at administrere varetagelse af digitale certifikater, herunder udstedelse
[5] DANE (DNS-based Authentication of Named Entities) er en standard, der anvender DNS til at hjælpe klienter med at validere et anvendt certifikat eller en certifikatudsteder. DANE forudsætter, at domæner er sikret med DNSSEC.

Hash funktioner
I forbindelse med en signering anvendes der en Hash-funktion[6]. Hash-funktioner anvendes til udregning af hash-værdier til brug ved generering af en digital signatur, ved generering af tilfældige tal og til bulk-kryptering i de situationer, hvor man ønsker at overføre en MAC[7]-værdi for de transmitterede data. Afhængig af hvilken anvendelse, der er tale om, vil forskellige Hash-funktioner kunne anbefales.

Anbefaling
For nøgleudveksling og certifikat-verifikation anbefales det, at man understøtter en eller flere af følgende hash-funktioner: SHA-512, SHA-384 eller SHA-256.

For Hash-funktioner til anvendelse i forbindelse med bulk-kryptering eller generering af tilfældige tal anbefales det, at man understøtter en eller flere af følgende funktioner: HMAC-SHA-512, HMAC-SHA-384 eller HMAC-SHA-256[8].

For alternative valgmuligheder henvises til bilag.

Øvrige parametre
For TLS vil der også være en lang række andre parametre, som har indflydelse på den samlede sikkerhed i en løsning. I det følgende er nævnt de sikkerhedsrelaterede parametre, der i de typiske standardkonfigurationer kan forringe den overordnede sikkerhed.

Komprimering
Anvendelsen af kompression som del af en TLS-session bør undgås. Baggrunden for dette er, at komprimering genererer mange gentagelser datastrømmen og dermed sårbarheder, som en angriber med en målrettet interaktion med serveren, kan udnytte til at forstå indholdet af de transmitterede informationer.

Genforhandling (renegotiation)
Funktionen genforhandling findes i tidligere versioner af TLS (før version 1.3), hvor det var muligt at initiere en ny genforhandling efter, at den indledende forhandling var afsluttet. Allerede i sit oprindelig design var denne metode usikker, hvorfor den blev erstattet af en ny metode. Den oprindelige blev derefter omdøbt til usikker genforhandling (”Insecure renegotiation”) mens den nye blev kaldt klient-initieret genforhandling (”Client initiated renegotiation”).

Anbefaling
Det anbefales, at genforhandling slås fra (værdien af sikkerhedsparameteren Renegotiation sættes til ”Off”), da genforhandling kan muliggøre potentielle DDOS-angreb.

0-RTT (zero Round-Trip Time)
I tidligere versioner af TLS krævede etableringsfasen to omgange kommunikation frem og tilbage mellem klient og server før applikations-fasen kunne etableres. I TLS 1.3 er dette skåret ned til en enkelt omgang. Ved anvendelse af 0-RTT parameteren kan man

[6] En hash-funktion er en matematisk envejsfunktion, der kan omsætte specifikke data til et unikt irreversibelt digitalt fingeraftryk.
[7] MAC Message Authentication Code.
[8] HMAC-SHA-xxx er en kompleks variant af SHA-xxx der sikrer et unikt resultat når hash-værdien af to sammenstillede ”tekster” skal udregnes.

tillade, at der allerede i etableringsfasen overføres applikationsdata. Dette sker dog på bekostning af sikkerheden, idet man herved bliver sårbar over for "replay"-angreb. Derfor anbefales det, at værdien af 0-RTT sættes til "Off".

Anbefaling

Det anbefales af værdien af 0-RTT sættes til "off".

OCSP hæftning (stapling)

TLS-klienten kan verificere validiteten af serverens (X.509) certifikat ved brug af OCSP (Online Certificate Status Protocol). Herved kan klienten, ved at kontakte serverens certifikatudsteder, få oplyst status på servercertifikatet ("godt", "tilbagekaldt" eller "ukendt"). Denne kommunikation er både tidskrævende, og medfører en risiko for brud på fortroligheden af private oplysninger[9]. OCSP-hæftning muliggør, at serveren selv kan fremsende dokumentation for gyldigheden af dens certifikat i form af en tidsstemplet signatur fra certifikatudbyderen. Herved fjernes klientens behov for at kontakte certifikatudstederen samtidig med, at risikoen for brud på fortroligheden i denne sammenhæng fjernes. Derfor anbefales det at værdien af OCSP hæftning sættes til "On".

Anbefaling

Det anbefales at værdien af OCSP-hæftning sættes til "On".

[9] Private oplysninger kan i denne sammenhæng være oplysninger om, hvilke web-sider man tilgår på nettet.

Referencer

IT Security Guidelines for Transport Layer Security (TLS) (23/5/2019)
https://english.ncsc.nl/publications/publications/2019/juni/01/it-security-guidelines-for-transport-layer-security-tls

Transport Layer Protection Cheat Sheet – OWASP
https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

Bilag: Oversigt over konfigurationsparametre

#	Emne for retningslinjer	Værdi	Status
1	TLS versioner	TLS 1.3 TLS 1.2	God
		TLS 1.1 TLS 1.0	Bør udfases
		SSL 3.0 SSL 2.0 SSL 1.0	Utilstrækkelig
2	Algoritmer for – Nøgleudveksling	ECDHE	God
		DHE	Tilstrækkelig
		RSA	Bør udfases
		DH25 ECDH27 KRB5 NULL PSK SRP	Utilstrækkelig
3	Algoritmer for – Certifikat verifikation	ECDSA RSA	God
		DSS24 EXPORT-variants PSK Anon NULL	Utilstrækkelig
4	Algoritmer for – Bulk kryptering	AES-256-GCM ChaCha20-Poly1305 AES-128-GCM	God
		AES-256-CBC AES-128-CBC	Tilstrækkelig
		3DES-CBC	Bør udfases
		AES-256-CCM_833 AES-128-CCM_832 IDEA DES RC4 NULL	Utilstrækkelig
5	Hash funktioner for – Nøgleudveksling	SHA-512 SHA-384 SHA-256	God
		Øvrige funktioner	Bør udfases

#	Emne for retningslinjer	Værdi	Status
6	Hash funktioner for – Certifikat verifikation	SHA-512 SHA-384 SHA-256	God
		SHA-1 MD5	Utilstrækkelig
7	Hash funktioner for – Bulk kryptering	HMAC-SHA-512 HMAC-SHA-384 HMAC-SHA-256	God
		HMAC-SHA-1	Tilstrækkelig
		HMAC-MD5	Utilstrækkelig
8	RSA Nøglelængder	Mindst 3072	God
		2048-3071	Tilstrækkelig
		Mindre end 2048	Utilstrækkelig
9	Understøttet elliptisk kurver	secp384r1 secp256r1 x448 x25519	God
		secp224r1	Bør udfases
		Andre kurver	Utilstrækkelig
10	Understøttet ”finite field” grupper	ffdhe4096 (RFC 7919) ffdhe3072 (RFC 7919)	Tilstrækkelig
		ffdhe2048 (RFC 7919)	Bør udfases
		Andre grupper	Utilstrækkelig
11	Komprimering	Ingen komprimering	God
		Application-level kompri-mering	Tilstrækkelig
		TLS-komprimering	Utilstrækkelig
12	Usikker genforhandling (Insecure renegotiation)	Off (eller N/A for TLS 1.3)	God
		ON	Utilstrækkelig
13	Klient-initierede genforhandling (Client initiated renegotiation)	Off (eller N/A for TLS 1.3)	God
		ON	Utilstrækkelig
14	0-RTT	Off (eller N/A før TLS 1.3)	God
		ON	Utilstrækkelig
15	OCSP hæftning (stapling)	ON	God
		Off	Tilstrækkelig