TRANSPORT LAYER SECURITY WHITEPAPER

LEAKFREE – IT Security made simple

Inhoudsopgave

Introductie **1**

Best Practices **4**

Certificaten 5

Protocolversies 8

Versleuteling 9

Configuratie 11

Samenvatting 15

Appendix **16**

Kwetsbaarheden in TLS 16

Referenties 19

Documentsgeschiedenis 21

INTRODUCTIE

Introductie

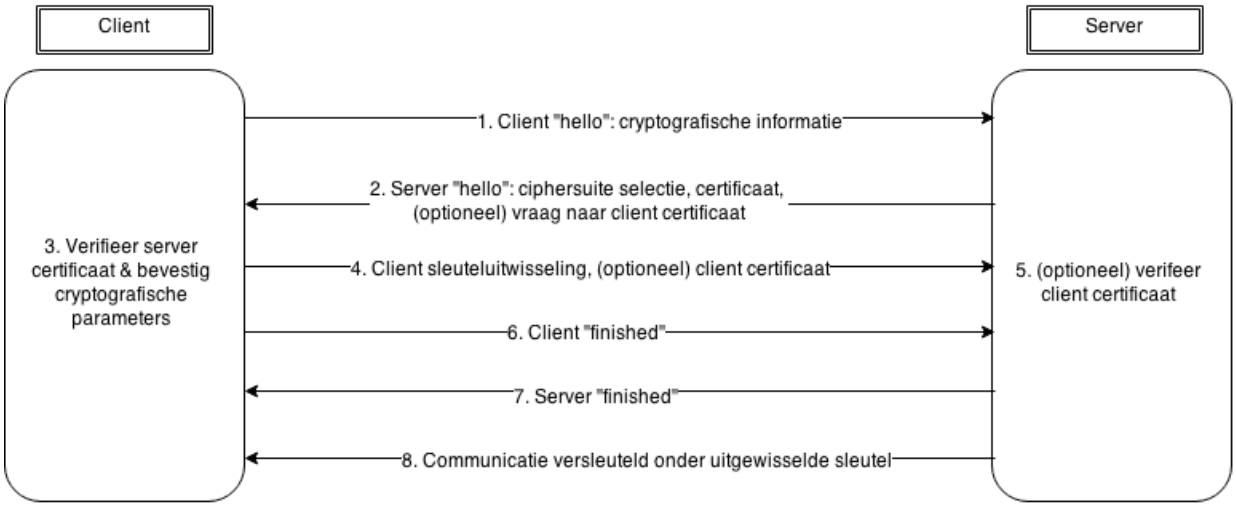
Transport Layer Security (TLS), vaak nog Secure Sockets Layer (SSL) genoemd, is een protocol dat cryptografisch beveiligde verbindingen tussen twee computersystemen faciliteert. TLS wordt voor een brede waaier aan verschillende toepassingen gebruikt: van web (HTTPS) en e-mail (STARTTLS voor IMAP, POP3 en SMTP) tot Voice-over-IP (VoIP) en Virtual Private Network (VPN) verkeer. TLS beschermt de communicatie tussen client en server om de vertrouwelijkheid en integriteit van gevoelige informatie te garanderen. TLS vormt hiermee de ruggengraat van veilige communicatie over internet en het gebruik van versleutelde verbindingen is in sommige gevallen dan ook verplicht binnen sommige organisaties en bedrijfssectoren.

Deze whitepaper heeft dan ook als doel een overzicht te bieden van de potentiele veiligheidsrisico's die bij TLS komen kijken en advies te bieden in de vorm van enkele *'best practices'* die zich met name richten op TLS-gebruik bij webservers (in de vorm van HTTPS) maar in veel gevallen generaliseerbaar zijn naar TLS-gebruik in het algemeen. De inhoud van deze whitepaper is technisch van aard maar niet softwarepakket-specifiek en dient vooral als hulpmiddel bij het evalueren van de veiligheid uw TLS configuratie.

INTRODUCTIE

HET TLS PROTOCOL

Een met TLS beveiligde verbinding bestaat uit twee losse fasen: de zogenaamde *handshakefase* en de *datauitwisselingsfase*. Via de handshake worden tussen client en server de details van de verbinding (zoals de protocolversie, de ciphersuite, een bewijs van identiteit dmv. certificaten) afgesproken. Tijdens de handshake controleert de client de authenticiteit van identiteit van de server om vast te stellen dat deze ook daadwerkelijk is welke ze claimt te zijn en kan de server eventueel (maar lang niet altijd) ook de identiteit van de client vaststellen. Voor het verifieren van deze certificaten is TLS afhankelijk van zogenaamde *Certificaat Autoriteiten (CAs)*, die de publieke sleutel van het certificaat van iedere dienst dienen te *signen*. Ook wordt tijdens de handshake via asymmetrische cryptografie een symmetrische *sessie-sleutel* afgesproken waarmee de rest van het verkeer versleuteld wordt, door middel van *bulkversleuteling,* tijdens de datauitwisselingsfase.



Schematische weergave van een TLS sessie

INTRODUCTIE

COMPLICATIES

Er bestaan op dit moment zes TLS versies. Drie onder de oude naam: Secure Sockets Layer (SSL) versies 1.0, 2.0 en 3.0, en drie onder de naam TLS: versies 1.0, 1.1 en de meest recente versie 1.2. Verschillende servers (zoals webservers) en clients (zoals webbrowsers) kunnen meerdere, onderling onverenigbare, versies ondersteunen. Ieder met eigen mogelijkheden en beperkingen op cryptografisch gebied. Een van de factoren die een goede TLS-configuratie compliceert is het feit dat deze niet alleen veilig maar ook compatibel moet zijn met de mogelijkheden van beoogde, mogelijk verouderde, clients. Om compatibel te zijn dient er ten minste één keuze (op het gebied van bijvoorbeeld TLS versie, *cipher suite* of sleutellengte) ondersteund te worden door de configuraties van zowel client als server. Oudere clienten ondersteunen niet altijd de nieuwste TLS versies en de veiligste cipher suites en sleutellengtes. Ook is het naleven van veiligheidsrichtlijnen van invloed op de *Public Key Infrastructure (PKI)* van uw organisatie. Niet iedere certificaatleverancier kan immers alle typen certificaten leveren.

De onoverzichtelijkheid die hiermee gepaard gaat leidt vaak tot kwetsbare configuraties. Zo is vaak te zien dat wanneer men kiest voor de breedst mogelijke compatibiliteit, dat ten koste gaat van de algehele veiligheid. Daarnaast is vaak te zien dat er tijdens installatie voor *'out-of-the-box'* configuraties wordt gekozen, waarna er in de loop der tijd weinig herevaluatie van inmiddels verouderde (en vaak kwetsbare) instellingen plaatsvindt.

In de afgelopen jaren zijn er verschillende kwetsbaarheden in zowel bepaalde versies van het TLS protocol als specifieke implementaties daarvan ontdekt [1,2,3] en hebben we gezien hoe serieus de gevolgen van aanvallen op zowel de onderliggende infrastructuur [4,5] als TLS-servers zelf [6, 7] kunnen zijn. Ook de resultaten van een recent door LeakFree uitgevoerde test van de TLS configuratie van diverse grote Nederlandse websites [8] laten zien dat de veiligheid hier op punten vaak te wensen overlaat. Het is daarom ook zaak om zowel *'best practices'* toe te passen als het controleren van TLS-configuraties tot een integraal onderdeel van uw reguliere audits te maken.

Best Practices

In deze sectie behandelen we de aanbevolen *best practices* voor verschillende onderdelen van uw TLS configuratie, alsmede de mogelijke risico's die een onveilige configuratie met zich mee kan brengen. Dit overzicht is uiteraard niet uitputtend maar dekt de belangrijkste en meest voorkomende issues.

Bij het implementeren en naleven van *best practices* is het van belang een goede afweging te maken tussen functionaliteit (met name compatibiliteit) en veiligheid. Zoals eerder vermeld kan het tussen deze twee zaken botsen en het is daarom van belang een goed overzicht van de gebruikersbasis van de systemen in kwestie te hebben. Wat voor clients dienen er te kunnen communiceren met uw servers? Hoeveel hiervan zijn naar alle waarschijnlijkheid verouderde clients die nieuwere, veiligere, protocolversies, *cipher suites* en sleutellengtes niet ondersteunen? Heeft u invloed op de clients en zo nee, is het de moeite waard steken te laten vallen op het gebied van veiligheid om een verouderd (en mogelijk slinkend) deel van uw gebruikersbasis te behouden?

Zorg er in ieder geval voor dat u op alle afzonderlijke vlakken in ieder geval aan de minimale eisen voldoet maar bij voorkeur ook aan de aanbevolen eisen. Daar waar cruciale compatibiliteit botst met de aanbevolen eisen wordt het aangeraden om op deze specifieke gebieden terug te schalen naar de minimale eisen. Mocht het toch niet mogelijk zijn om aan de minimale eisen te voldoen dan mag dit echter geen reden zijn om TLS volledig uit te schakelen, streef naar het hoogst haalbare. Houdt in het algemeen de vuistregel aan: '*sta geen zwakkere instellingen toe dan absoluut noodzakelijk*'.

De aanbevolen best practices zijn gebaseerd op een combinatie van autoratieve bronnen [9, 10, 11] en onze eigen ervaringen. Uiteraard zijn niet alle configuraties die zich niet strict aan de aanbevelingen houden onveilig en het kan in uitzonderlijke situaties, vanwege bijvoorbeeld efficientie of compatibiliteitseisen, noodzakelijk zijn hiervan af te wijken. Het goed instellen en beoordelen van de veiligheid van een TLS configuratie vereist specialistische kennis en contact met een security professional wordt dan ook aangeraden.

AANBEVELINGEN

- *Signing*: Zorg dat het certificaat in kwestie ondertekend, ofwel *gesigned,* is door een vertrouwde *CA*, uitgekozen in overleg met de beheerders van uw *PKI*. Het genereren van een *hash* van deze *signature* dient te gebeuren met een cryptografisch sterk *hashing algoritme* zoals SHA-256, SHA-384 of SHA-512. SHA-1 is een nog steeds veel gebruikt *hashing algoritme* maar dit wordt vanwege veiligheidsproblemen langzaam uitgefaseerd. Mocht u een sterk verouderde gebruikersbasis hebben (zoals gebruikers van Internet Explorer 6 op Windows XP) is dit echter de veiligste optie. In ieder geval dienen MD2, MD4 en MD5 ten alle tijde vermeden te worden.

- *Certificaatverificatie*: De certificaatverificatiemethode dient bij voorkeur ECDSA (Elliptic Curve Digital Signature Algorithm) te zijn. Wanneer dit compatibiliteitsproblemen oplevert is RSA ook een veilige keuze. In het geval van RSA wordt aangeraden om sleutels met een lengte van minimaal 2048-bits, en bij voorkeur 3072-bits, te kiezen. Hoewel men soms nog sleutels met een lengte van 1024-bits aanraadt betekent de almaar toenemende rekenkracht van moderne computersystemen dat sleutellengtes hiermee moeten meeschalen. In het geval van ECDSA wordt een sleutellengte van minimaal 224-bits en bij voorkeur 256-bits aangeraden.

- *Volledige hostname dekking*: Zorg verder ook dat het certificaat de hostname(s) van de beoogde server(s) dekt, een certificaat voor www.voorbeeld.nl is immers geen certificaat voor webmail.voorbeeld.nl of voorbeeld.nl.

- *Bescherm uw private keys:* De veiligheid van uw certificaat valt of staat met de veiligheid van de bijbehorende *private keys*. Limiteer toegang tot de private keys tot de kleinst mogelijke, strict noodzakelijke, groep medewerkers en sla ze uitsluitend op veilige systemen op. Maak gebruik van wachtwoord-bescherming voor de sleutels op backups (in productieomgevingen heeft dit weinig zin) om te voorkomen dat ze in verkeerde handen vallen. Hernieuw certificaten met een vast interval (bij voorkeur ieder jaar) en altijd met nieuwe sleutels. Mochten de sleutels onverhoops

toch in verkeerde handen vallen, *revoke* de oude certificaten dan onmiddelijk en genereer nieuwe sleutels.

- *Denk aan de gehele certificate chain*: Niet alleen het certificaat van de server in kwestie dient in orde te zijn maar de gehele bovenliggende *certificate chain*. In veel gevallen is het namelijk nodig om twee of meer certificaten te hebben voor een complete vertrouwensketen (zoals een organisatiebreed certificaat en een specifiek certificaat voor een interne server). Het niet periodiek vernieuwen, laten verlopen of in het algemeen incorrect configureren van certificaten in de bovenliggende *certificate chain* maakt het certificaat van de server in kwestie zwakker of zelfs ongeldig.

- *Overig*: Tevens dient men sleutels gegenereerd met OpenSSL versies 0.9.8c-1 tot en met 0.9.8g-9 op een Debian-systeem te vermijden vanwege een zwakheid in het onderliggende sleutelgeneratieproces [12].

RISICO'S

- *Ongeldige certificaten*: Verlopen certificaten dienen ten alle tijde vermeden te worden. Het geldigheidsinterval van het certificaat dient in ieder geval de nabije toekomst te dekken. Een ongeldig certificaat biedt geen enkele veiligheid.

- *Self-signed certificaten*: Vertrouw niet op zogenaamde *self-signed* certificaten. Het wil nogal eens voorkomen dat men denkt kosten te kunnen besparen door certificaten zelf te *signen* in plaats van dat te laten doen door een CA. Of het nu gaat om extern toegankelijke servers of alleen intern benaderbare servers, het veilig gebruiken van *self-signed* certificaten brengt enorme verborgen kosten met zich mee [13] en gaat al snel mis.

- *Onvolledige hostname dekking*: Wanneer niet alle hostnames geassocieerd met een server gedekt worden door het certificaat is het mogelijk dat clients die met de server proberen te communiceren via een ongedekte hostname onbeveiligd communiceren.

BEST PRACTICES - CERTIFICATEN

- *Onveilige certificaatverificatie algoritmen of sleutellengtes*: Wanneer men kiest voor een onveilig certificaatverificatie algoritme of onveilige bijbehorende sleutellengtes is het mogelijk voor een aanvaller om een vals certificaat te produceren en dat te gebruiken in een *Man-In-The-Middle* aanval.

- *Onveilige signature hashing algoritmen*: Als een certificaat gesigned is met een zwak *hashing algoritme*, zoals bijvoorbeeld MD5, is het mogelijk voor een aanvaller om door middel van een zogenaamde *collision attack* een eigen certificaat te genereren dat door gebruikers als het originele zal worden geinterpreteerd [14].

- *Zwakke certificate chain*: Al het bovenstaande is uiteraard ook van toepassing op de bovenliggende certificaten in de *certificate chain*. Certificaten waarvan de *certificate chain* niet in orde is dienen ten alle tijde vermeden te worden aangezien de veiligheid van een certificaat slechts zo sterk is als die van het zwakste certificaat in de *chain*.

- *Onbeschermde private keys*: De veiligheid van een certificaat valt of staat met de veiligheid van de bijbehorende *private keys*. Als deze in verkeerde handen vallen komt daarmee de veiligheid van het certificaat en alle toekomstige (en in sommige gevallen eerdere) versleutelde verbindingen die daarvan gebruik maken in geding.

- *Overig:* Certificaatsleutels dienen nooit gegenereerd te zijn met OpenSSL versies 0.9.8c-1 tot en met 0.9.8g-9 op een Debian-systeem [12]. Sleutels gegenereerd met dit pakket hebben een voorspelbare *private key* waardoor aanvallers gemakkelijk een *Man-In-The*-Middle aanval kunnen uitvoeren en versleuteld verkeer kunnen onderscheppen.

BEST PRACTICES - PROTOCOLVERSIES

AANBEVELINGEN

De nieuwste versie van TLS is versie 1.2. Met betrekking tot de verschillende versies (SSL 1.0 is vanwege veiligheidsproblemen nooit in gebruik genomen) geldt het volgende:

1. SSL 2.0 is onveilig en dient uitgeschakeld te worden.
2. SSL 3.0 is onveilig en dient uitgeschakeld te worden.
3. TLS 1.0 heeft meerdere issues en kan, met heel zorgvuldige configuratie, veilig gemaakt worden.
4. TLS 1.1 en 1.2 zijn in sommige gevallen kwetsbaar maar kunnen met zorgvuldige configuratie veilig gemaakt worden.

Zorg dat TLS 1.2 in ieder geval, als primair protocol, ondersteunt wordt. Mocht het nodig zijn om compatibiliteit met oudere clients te behouden, ondersteun dan TLS 1.0 en 1.1 alleen zolang dit nodig is.

RISICO'S

- *SSL 2.0* is kwetsbaar voor een *Man-In-The-Middle* aanval [15].

- *SSL 3.0* is kwetsbaar voor meerdere aanvallen, waaronder POODLE [1] en BEAST [16].

- *TLS 1.0* is kwetsbaar voor meerdere aanvallen, waaronder BEAST [16] en in sommige gevallen POODLE [1].

- *TLS 1.1* en 1.2 zijn in sommige gevallen kwetsbaar voor POODLE [1].

BEST PRACTICES - VERSLEUTELING

AANBEVELINGEN

- *Sleuteluitwisseling*: De sleuteluitwisselingsmethode dient bij voorkeur ECDHE (Elliptic Curve Diffie-Hellman Ephemeral) te zijn om *forward secrecy* te garanderen. Wanneer dit compatibiliteitsproblemen oplevert is RSA ook een veilige keuze. In het geval van RSA wordt aangeraden om sleutels met een lengte van minimaal 2048-bits, en bij voorkeur 3072-bits, te kiezen. In het geval van ECDHE wordt een parameterlengte van minimaal 224-bits en bij voorkeur 256-bits aangeraden.

- *Bulkversleuteling*: Bulkversleuteling dient bij voorkeur te gebeuren met AES (Advanced Encryption Standard) in GCM in een AEAD (Authenticated Encryption with Associated Data) mode zoals bijvoorbeeld GCM (Galois/Counter Mode). Vermijd in ieder geval *blockciphers* (zoals AES) in CBC (Cipher-Block Chaining) mode en schakel alle ondersteuning voor de algoritmes DES, RC2 en RC4 uit.

  Hoewel RC4 in het verleden vaak aangeraden werd als 'workaround' voor aanvallen op *blockciphers* in CBC mode kampt het met enkele zwakheden [28] waardoor ondersteuning alleen toelaatbaar is vanwege onoverkomelijke compatibiliteitsissues en dan alleen binnen de context van TLS 1.0 en als laatste voorkeursoptie. Een sleutellengte van minimaal 128-bits, en bij voorkeur 256-bits, wordt voor ieder algoritme aangeraden. Bijvoorbeeld: AES-256-GCM of AES-128-GCM.

- *Export varianten*: Export varianten van algoritmen (vaak aangeduid met het voorvoegsel EXP-), inclusief die van anders als veilig beschouwde algoritmen, dienen ten alle tijde vermeden te worden aangezien het hier voor export-reguleringen afgezwakte versies betreft.

- *Attentie!* Een keuze voor compatibiliteit met oudere clients kan tot onoverkomelijke veiligheidsproblemen leiden. Mocht men toch ondersteuning voor TLS 1.0 of SSL 3.0 willen blijven garanderen, dan kan dat alleen als men hier blockciphers in CBC mode vermijd. Aangezien andere modes in die protocolversies geen optie zijn blijft alleen het kwetsbare RC4 over als alternatief [17]. Ondersteuning voor TLS 1.0 betekent

dan ook de keuze tussen kwetsbaarheid voor aanvallen op RC4 of kwetsbaarheid voor de BEAST aanval. Aangezien de meeste moderne browsers ingebouwde bescherming tegen BEAST hebben is over het algemeen ondersteuning voor blockciphers in CBC mode het verstandigst. Dit ligt echter anders als u een gebruikersbasis met sterk verouderde browsers heeft, waardoor BEAST een reelere bedreiging kan zijn dan aanvallen op RC4. Een juiste afweging vereist een goed overzicht van uw infrastructuur, gebruikersbasis en dreigingsmodel.

RISICO'S

- *Onveilige of afgeraden algoritmen of sleutellengtes voor sleuteluitwisseling en bulkversleuteling*: Als de gebruikte versleuteling (ofwel door de keuze voor een onveilig algoritme, ofwel door een onveilige sleutellengte) van een verbinding niet sterk genoeg is kan een aanvaller onderschepte, versleutelde communicatie in een later stadium kraken om zo de inhoud te bemachtigen.

  Daarnaast stelt gebruik van onveilige sleuteluitwisselingsmethoden een aanvaller in staat tot het uitvoeren van een *Man-In-The-Middle* aanval.

- *Onveilige blockcipher modus*: Het gebruik van anderszins veilige *blockciphers* in CBC mode kan uw configuratie kwetsbaar maken voor verschillende aanvallen, waaronder BEAST [16], LUCKY 13 [27] en POODLE [1].

BEST PRACTICES - CONFIGURATIE

AANBEVELINGEN

- *Up-to-date TLS library*: Wees op de hoogte van welke TLS library of libraries uw server gebruikt voor diverse diensten en zorg dat deze allemaal regelmatig geupdate worden.

- *Secure Server-initiated Session Renegotiation*: TLS maakt het mogelijk voor clients en servers om datauitwisseling stop te zetten om opnieuw af te spreken hoe de data versleuteld wordt. Het kan noodzakelijk zijn voor servers om *renegotiation* te kunnen initieren maar er is geen reden voor clients om dat te doen. Mocht men renegotation willen ondersteunen zorg dan dat zogenaamde *insecure renegotiation* uit staat om aanvallen waarbij HTTP requests worden geinjecteerd te voorkomen.

- *Forward Secrecy*: Forward Secrecy maakt het mogelijk om de vertrouwelijkheid van mogelijk onderschepte TLS-communicatie te blijven garanderen ook nadat de *private key* van het bijbehorende certificaat in verkeerde handen terecht is gekomen. Dit gebeurt door niet direct gebruik van de sleutels van client en server te maken maar van een zogenaamde *ephemeral* sleutel die alleen van toepassing is op de communicatiesessie in kwestie. Na afloop van de communicatie wordt deze sleutel gewist en op basis van de *private key* van het certificaat valt deze *ephemeral* sleutel niet te achterhalen. Het gebruik van ECDHE als algoritme voor sleuteluitwisseling garandeert forward secrecy.

- *HTTP Strict Transport Security (HSTS)*: Ondanks het feit dat webpagina's via HTTPS over een TLS verbinding worden geladen kan het zo zijn dat sommige elementen onveilig geladen worden, ofwel door de architectuur van de webapplicatie ofwel door een zogenaamde *SSL-stripping* aanval [18]. Om dit te voorkomen kunnen servers (die daar ondersteuning voor bieden) geconfigureerd worden met de HSTS optie die afdwingt dat alle interactie tussen server en client via HTTPS plaatsvind en nooit over HTTP.

- *Compressie*: Compressiegebruik kan aanvallers informatie verschaffen over de versleutelde communicatie. Er zijn, binnen de context van TLS, twee soorten

compressie om rekening mee te houden: TLS-compressie en applicatiespecifieke compressie. In het eerste geval verdient het de voorkeur om deze volledig uit te schakelen (aangezien ze zelden gebruikt wordt). In het geval van applicatiespecifieke compressie (zoals bijvoorbeeld HTTP compressie via gzip, deflate of compress) is dit lastiger tegen te gaan. Deze compressie is vaak vereist voor efficient applicatiegebruik en verdient dan ook een gevalsspecifieke afwezing. Het tegengaan kan hier wijzigingen in de applicatiecode vereisen. Mochten zaken zoals efficientie en bandbreedte geen issue zijn (of secundair aan de veiligheid) dan verdient het de voorkeur applicatiespecifieke compressie uit te zetten.

- *Cookies*: Cookies dienen door de webapplicatie in kwestie als 'secure' gemarkeerd te worden om te voorkomen dat ze onveilig verstuurd worden met alle gevolgen van dien.

- *OCSP Stapling*: OCSP stelt een client in staat om de geldigheid van een aangeboden server-certificaat na te vragen bij de certificaatleverancier via het OCSP-protocol. Om te voorkomen dat privacy-gevoelige informatie (zoals welke clients er communiceren met uw servers) terecht komen bij derde partijen als certificaatleveranciers verdient het voorkeur om OCSP-stapling te ondersteunen, waarbij de server zelf de OCSP informatie verstrekt.

- *Protocol downgrade attack bescherming*: Bepaalde recente versies van TLS implementaties (zoals de bijvoorbeeld nieuwste OpenSSL versie) bieden ondersteuning voor de TLS_FALLBACK_SCSV optie. Deze optie stelt clients in staat om expliciet aan de server te vermelden dat men probeert de TLS verbinding met een lagere protocolversie te herinitieeren na een eerdere poging met een hogere versie, iets wat voorkomt tijdens bepaalde aanvallen op TLS [1]. Dit stelt de server instaat om de hoogst ondersteunde versie te vergelijken met de door de client aangegeven versie in de handshake. Als de versie van de client lager is kan de server vervolgens de connectie weigeren omdat de server weet dat de client een hogere versie ondersteunt dan waar met de fallback naar gevraagd wordt. Ondersteuning

van TLS_FALLBACK_SCSV als remedie tegen bepaalde aanvallen op TLS verdient dan ook de voorkeur.

- *Session resumption*: Afhankelijk van de configuratie van uw server verdient het de voorkeur session resumption (zowel op basis van session IDs als TLS tickets) uit te schakelen.

RISICO'S

- *Out-of-date TLS library*: Als uw TLS library niet up-to-date is kan uw server daarmee blootgesteld worden aan allerlei kwetsbaarheden zoals HEARTBLEED [2] of CVE-2014-0224 [3]. De impact hiervan kan varieeren van het mogelijk maken van *Man-In-The-Middle* aanvallen tot overname van de server.

- *Insecure session renegotiation*: Als men client-initiated session renegotiation toestaat wordt de server daarmee kwetsbaar voor zogenaamde *Denial-of-Service (DoS)* aanvallen [19]. Als men insecure session renegotiation toestaat stelt dat een aanvaller in staat om een *Man-In-The-Middle* aanval uit te voeren waarbij HTTP requests en andere data geinjecteerd kunnen worden, met ruwweg dezelfde gevolgen als een *Cross-Site Request Forgery (CSRF)* aanval [20].

- *Geen Forward Secrecy*: Als men geen ondersteuning biedt voor Forward Secrecy, dan kan een aanvaller na diefstal van de *private key* van het certificaat meteen alle eerder onderschepte data met terugwerkende kracht ook ontsleutelen. Dit betekent dat na een inbraak niet alleen alle toekomstige maar ook alle eerdere communicatie in handen van de aanvaller kan vallen.

- *Geen HTTP Strict Transport Security (HSTS)*: Als men geen ondersteuning biedt voor HSTS dan kan een aanvaller door middel van een zogenaamde *SSL-stripping* aanval er voor zorgen dat er nooit een versleutelde sessie wordt opgezet maar dat alle communicatie onversleuteld verloopt.

- *TLS-Compressie*: Als men ondersteuning biedt voor TLS-compressie kan uw server daarmee kwetsbaar worden voor de CRIME [21] aanval.

- *HTTP-Compressie*: Als u HTTP-compressie ondersteunt kan uw server daarmee kwetsbaar worden voor de BREACH [22] aanval.

- *Onveilige cookies*: Cookies die niet als *secure* gemarkeerd zijn [23] kunnen onderschept worden door een aanvaller die daarmee de bijbehorende sessie kan kapen.

- *Session resumption*: Servers die *client-certificate authentication* ondersteunen, RSA of (EC)DHE sleuteluitwisseling gebruiken en session resumption toestaan zijn mogelijk kwetsbaar voor de TRIPLE HANDSHAKE [24] aanval.

BEST PRACTICES - SAMENVATTING

SAMENVATTING

Hieronder vindt u, per categorie, een kortbondig overzicht van de bovenstaande adviezen onderverdeeld in de klassen *aanbevolen, voldoende* en *legacy*. Waarbij de keuze tussen aanbevolen en voldoende afhankelijk is van reguliere compatibiliteitsissues en de keuze voor legacy alleen te rechtvaardigen is vanwege onoverkomelijke compatibiliteitsissues.

	Aanbevolen	**Voldoende**	**Legacy**
Signature hashing	SHA-512	SHA-384, SHA-256	SHA-1
Certificaatverificatie	ECDSA-256, RSA-3072	ECDSA-224, RSA-2048	N/A
Sleuteluitwisseling	ECDHE-256	ECDHE-224, ECDH-256, ECDH-224, RSA-3072, RSA-2048	N/A
Bulkversleuteling	AES-256-GCM	AES-128-GCM	AES-256-CBC, AES-128-CBC

	Aanbevolen instelling
SSL 2.0	Uit
SSL 3.0	Uit
TLS 1.0	Alleen legacy
TLS 1.1	Aan
TLS 1.2	Aan
Client-initiated renegotiation	Uit
Insecure renegotiation	Uit
TLS Compression	Uit
HTTP Compression	Uit wanneer mogelijk
HSTS	Aan
Secure cookies	Aan
OCSP Stapling	Aan
TLS_FALLBACK_SCSV	Aan

APPENDIX: KWETSBAARHEDEN IN TLS

Kwestbaarheden in TLS

Deze sectie bevat een kort overzicht van de meest prominente kwetsbaarheden die van toepassing zijn op TLS. Specifieke zwakheden in onveilige versleutelingsalgoritmen worden hier niet behandeld. Sommigen van de kwetsbaarheden zijn intrinsiek aan het ontwerp van TLS, anderen zijn implementatie-specifiek.

BEAST

BEAST [16] is een kwetsbaarheid bij gebruik van enkele *blockciphers* in *CBC modus* binnen TLS die een *Man-In-The-Middle* aanvaller in staat stelt om (delen van) versleutelde communicatie te achterhalen.

Gebruik van *blockciphers* in een zogenaamde *AEAD* modus (zoals *GCM* modus) is niet kwetsbaar voor BEAST. Tevens zijn *streamciphers* niet kwetsbaar voor BEAST maar wordt gebruik van RC4, de populairste *streamcipher* binnen TLS, afgeraden wegens andere zwakheden.

SSL 3.0 en TLS 1.0 zijn kwetsbaar voor BEAST, hoewel moderne browsers hier bescherming tegen hebben geintegreerd. Oudere browsers en browserversies hebben dat echter niet en communicatie tussen client en server waar geen van beiden bescherming tegen BEAST biedt is dan ook kwetsbaar.

TLS versies 1.1 en hoger zijn niet kwetsbaar voor BEAST.

CRIME

CRIME [21] is een kwetsbaarheid in TLS die misbruikt maakt van informatie die gelekt wordt door gebruik van TLS-compressie om een *Man-In-The-Middle* aanvaller in staat te stellen om (delen van) versleutelde communicatie te achterhalen.

Servers met ondersteuning voor TLS-compressie zijn kwetsbaar voor CRIME.

TIME

TIME [26] is een kwetsbaarheid in TLS en in bepaalde opzichten een verfijning van CRIME waarbij niet langer misbruik gemaakt wordt van door TLS-compressie gelekte informatie maar van informatie gelekt door de tijd die het duurt voor een webserver om requests te beantwoorden. In het geval van TIME hoeft de aanvaller ook niet langer een *Man-In-The-Middle* aanval uit te voeren maar is het dwingen van een doelwit-browser (door middel van

javascript verstopt in kwaadaardige webadvertenties bijvoorbeeld) tot het uitvoeren van grote hoeveelheden requests genoeg.

Aangezien TIME niet direct een zwakheid in TLS misbruikt is hier ook alleen tegen te wapenen door zorgvuldige en nauwkeurige ontwikkeling van de (web)applicaties op uw server.

BREACH

BREACH [22] is een kwetsbaarheid in TLS en ook een verfijning van CRIME waarbij geen misbruik gemaakt wordt van TLS-compressie maar van HTTP-compressie om een *Man-In-The-Middle* aanvaller in staat te stellen om (delen van) versleutelde communicatie te achterhalen.

Aangezien HTTP-compressie voor de meeste webservers een noodzaak is vanuit efficientie oogpunt is er geen volledige oplossing. Wel is het mogelijk om met erg zorgvuldig ontwerp van uw (web)applicaties de aanval tegen te gaan.

LUCKY 13

LUCKY 13 [27] is een kwetsbaarheid bij gebruik van enkele *blockciphers* in *CBC modus* binnen TLS die een *Man-In-The-Middle* aanvaller in staat stelt om (delen van) versleutelde communicatie te achterhalen.

Gebruik van *blockciphers* in een zogenaamde *AEAD* modus (zoals *GCM* modus) is niet kwetsbaar voor LUCKY 13. Tevens kan kwetsbaarheid voor LUCKY 13 vermeden worden door zorgvuldig ontwerp van (web)applicaties.

POODLE

POODLE [1] is een kwetsbaarheid bij het gebruik van *blockciphers* in *CBC modus* in SSL 3.0 en sommige implementaties van TLS versies 1.0, 1.1 en 1.2 TLS die een *Man-In-The-Middle* aanvaller in staat stelt om (delen van) versleutelde communicatie te achterhalen.

Gebruik van *blockciphers* in een zogenaamde *AEAD* modus (zoals *GCM* modus) is niet kwetsbaar POODLE. Tevens kunnen de ergste gevolgen van POODLE worden tegengegaan door SSL 3.0 uit te schakelen en TLS_FALLBACK_SCSV te ondersteunen.

APPENDIX: KWETSBAARHEDEN IN TLS

HEARTBLEED

HEARTBLEED [2] is een kwetsbaarheid in OpenSSL versies 1.0.1 eerder dan 1.0.1g die een aanvaller in staat stelt om gevoelige informatie uit het processgeheugen van de server te extraheren.

Door te zorgen dat u de nieuwste versie van OpenSSL heeft bent u beschermt tegen HEARTBLEED.

TRIPLE HANDSHAKE

TRIPLE HANDSHAKE [24] is een kwetsbaarheid in het session resumption mechanisme van TLS wanneer dit gebruikt wordt in combinatie met *client-certificate authentication* die een aanvaller in staat stelt om een *Man-In-The-Middle* aanval uit te voeren.

Er is nog geen complete fix voor TRIPLE HANDSHAKE maar in de tussentijd kan deze worden tegengegaan door session resumption uit te schakelen (wat echter wel een negatieve performance impact heeft) wanneer dat mogelijk is.

CVE-2014-0224

CVE-2014-0224 [3] is een kwetsbaarheid in OpenSSL versies voor 0.9.8za, versies 1.0.0 voor 1.0.0m en versies 1.0.1 voor 1.0.1h die misbruikt maakt van de manier waarop omgegaan wordt met zogenaamde ChangeCipherSpec berichten waardoor een aanvaller in staat is om een *Man-In-The-Middle* aanval uit te voeren.

Door te zorgen dat u de nieuwste versie van OpenSSL heeft bent u beschermt tegen CVE-2014-0224

APPENDIX: REFERENTIES

Referenties

1. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566
2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
3. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224
4. http://nl.wikipedia.org/wiki/Hack_bij_DigiNotar
5. http://hackmageddon.com/2011/12/10/another-certification-authority-breached-the-12th/
6. http://www.cbc.ca/news/business/heartbleed-bug-rcmp-asked-revenue-canada-to-delay-news-of-sin-thefts-1.2609192
7. http://www.reuters.com/article/2014/08/20/us-community-health-cybersecurity-idUSKBN0GK0H420140820
8. https://www.leakfree.nl/files/leakfree_ssl_tls_onderzoek.pdf
9. https://www.ncsc.nl/binaries/content/documents/ncsc-nl/dienstverlening/expertise-advies/kennisdeling/whitepapers/ict-beveiligingsrichtlijnen-voor-transport-layer-security-tls/1/ICT%2Bbeveiligingsrichtlijnen%2Bvoor%2BTransport%2BLayer%2BSecurity%2B%2BTLS%2B%2B%2Bleesversie%2B.pdf
10. https://www.ssllabs.com/
11. https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014/at_download/fullReport
12. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-0166
13. https://www.networking4all.com/nl/helpdesk/downloads/?a=f&r=1b23f1c95722adf4db80e9f4a06bd224
14. https://www.trailofbits.com/resources/flame-md5.pdf
15. http://www.sans.org/reading-room/whitepapers/threats/ssl-man-in-the-middle-attacks-480
16. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3389
17. https://community.qualys.com/blogs/securitylabs/2013/03/19/rc4-in-tls-is-broken-now-what
18. http://www.thoughtcrime.org/software/sslstrip/

APPENDIX: REFERENTIES

19. https://community.qualys.com/blogs/securitylabs/2011/10/31/tls-renegotiation-and-denial-of-service-attacks
20. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555
21. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-4929
22. https://media.blackhat.com/us-13/US-13-Prado-SSL-Gone-in-30-seconds-A-BREACH-beyond-CRIME-Slides.pdf
23. https://www.owasp.org/index.php/SecureFlag
24. https://secure-resumption.com/
25. https://www.isecpartners.com/media/106031/ssl_attacks_survey.pdf
26. https://www.owasp.org/images/e/eb/A_Perfect_CRIME_TIME_Will_Tell_-_Tal_Beery.pdf
27. http://www.isg.rhul.ac.uk/tls/Lucky13.html
28. http://www.isg.rhul.ac.uk/tls/

APPENDIX: DOCUMENTSGESCHIEDENIS

VERSIE 1.0

Wijziging	Datum	Auteur
Publicatie 1e versie	29-01-2015	Jos Wetzels

Over LEAKFREE

LeakFree is een jong IT-security consultancy bedrijf. Wij verzorgen onder andere beveiligingstesten en brengen whitepapers en beveiligingsadviezen uit. Hiermee helpen wij onze opdrachtgevers risico's en potentiële bedreigingen voor hun digitale infrastructuur en gevoelige informatie te identificeren, zodat pro-actieve maatregelen genomen kunnen worden om toekomstige schade te voorkomen.

Onze medewerkers hebben jaren ervaring in de IT-security en verzorgden beveiligingstesten in diverse sectoren, van hostingbedrijven tot grote Nederlandse banken. Door haar kleinschalige en flexibele organisatie is LeakFree in staat kwalitatief goed en scherp geprijsd maatwerk te leveren in nauw overleg met haar opdrachtgevers.

CONTACT

E-mail: contact@leakfree.nl
http://www.leakfree.nl