HTTPS

**Inleiding**

Op deze website willen we ingaan op HTTPS. Alle ins en outs zullen we hier bespreken.

De s in https staat voor secure. Deze toevoeging aan het http protocol is gemaakt om gegevens veilig te verzenden over het internet. Standaard wordt http verkeer namelijk als plain tekst verzonden, als dit verkeer onderschept wordt kan iedereen de inhoud er van zomaar bekijken, bij privacygevoelige informatie is dit natuurlijk niet gewenst. Neem bijvoorbeeld het verzenden van je pincode naar de bank als je aan het internetbankieren bent, de pincode mag natuurlijk niet zomaar onderschept worden door een andere partij. Https is geen nieuw protocol en daarmee een opvolger van http, nee http legt de verbinding over SSL/TLS. Dit zorgt ervoor dat de verbinding geencrypt wordt, als de informatie die verzonden wordt dan onderschept wordt is deze informatie niet te lezen voor de derde partij. Hoe deze encryptie gedaan wordt en wat er nog meer allemaal met dit onderwerp te maken heeft willen we op deze website behandelen. Dit hebben we gesplitst in de volgende hoofdstukken:

* Hoofdstuk 1: we gaan hier in op het ontstaan van SSL/TLS, verschillende versies en ontwikkelingen.
* Hoofdstuk 2: we gaan hier in op de certificaten, hoe werkt dit?
* Hoofdstuk 3: we gaan hier in op: ……
* Hoofdstuk 4: we gaan hier in op: ……
* Hoofdstuk 5: we gaan hier in op: ……
* Hoofdstuk 6: we gaan hier in op: ……
* Hoofdstuk 7: we gaan hier in op: ……
* Hoofdstuk 8: we gaan hier in op: ……

**Ontstaan** SSL/TLS

SSL 1.0 is ontwikkeld door Netscape Communications in 1994, dit werd gedaan om gegevens tussen client en server applicaties over een onveilig netwerk, zoals het internet, veilig te verzenden. Deze versie van SSL was alleen toegankelijk binnen het eigen Netscape netwerk en had nog al wat mankementen, zo werd er geen volledige integriteit gewaarborgd van de gegevens. Ook waren er geen volgorde nummers of checksums om te controleren of er geen verkeerde data pakketjes werden meegestuurd over de verbinding(replay attacks). Later zijn deze mankementen wel deels opgelost door de ontwikkelaars van SSL 1.0. Maar korte tijd later, in 1995, werd SSL 2.0 uitgebracht en waren deze problemen geheel opgelost, ook werd er in SSL 2.0 gebruik gemaakt van one-way hashing, MD5 werd geïmplementeerd, op dat moment was MD5 nog veilig. SSL 2.0 werd wel op de markt gebracht in tegenstelling tot SSL 1.0 dat alleen binnen het eigen netwerk van Netscape Communications werd gebruikt. Aan SSL 2.0 zaten echter ook genoeg mankementen, vooral cryptografische zwakheden kwamen al snel aan het licht (nummer 4 van bronnen) . Voorbeelden zijn:

* Al in 1996 werd ontdekt dat MD5 niet geheel veilig was en werd er aangeraden gebruik te maken van andere algoritmen zoals SHA-1, MD5 was toen echter nog niet gekraakt.
* Beperkt gebruik van SSL Certificaten
* SSL 2.0 gebruikt dezelfde keys voor encryptie en authenticatie van authenticatie, dit gaf problemen bij verschillende ciphers. Ook waren de gebruikte keys slechts 40 bits.
* SSL 2.0 was kwetsbaar voor man-in-the-middle aanvallen, het netwerkverkeer wordt hierbij onderschept zonder dat de verzendende of ontvangende partijen er weet van hebben.

Als reactie op de problemen van SSL 2.0 werd al snel SSL 3.0 ontwikkeld, SSL 3.0 was een complete herziening van het protocol en veel van hiervoor genoemde problemen werden verholpen.

TLS 1.0 is een update voor SSL 3.0, TLS is de opvolger van SSL, de afkorting TSL staat voor Transport Security Layer, deze update vond plaats in 1996 en is niet heel vergaand. SSL 3.1 is dus vergelijkbaar met TLS 1.0. Deze naamsverandering is gekomen toen SSL werd overgedragen aan het IETF door Netscape. In 2006 werd TLS 1.1 uitgebracht en in 2008 TLS 1.2. Op dit moment, maart 2015 is er een concept versie van TLS 1.3. Om de ontwikkelingen goed te kunnen volgens is het nodig dat we eerst technisch wat uitleg geven over de werking van SSL/TLS.

TLS 1.2 is de beste verbindingsmogelijkheid die er op dit moment bestaat, TLS 1.1 is ook nog niet gekraakt maar TLS 1.2 heeft toch weer veel verbeteringen.

**Certificaten**

Voor het gebruik van SSL/TLS is een SSL-certificaat nodig, een dergelijk certificaat is aan te vragen bij een CA (Certificate Authority). SSL-certificaat is eigenlijk geen goede benaming omdat SSL verouderd is en de verbindingen over TLS gaan, maar de benaming SSL-certificaat is ingeburgerd vandaar dat er overal nog over SSL-certificaat wordt gesproken. Het doel van een certificaat is de identiteit van de website en de daarvoor opgezet https verbinding te verifiëren. Zo ben je er zeker van dat er nooit gecommuniceerd wordt met een andere (verkeerde) server.

Een SSL-certificaat heeft een bepaalde geldigheidsduur, het certificaat moet dus steeds weer vernieuwd worden om een veilige verbinding te garanderen.

Een SSL-certificaat is niet meer dan een tekstbestand met daarin bepaalde informatie, het hele principe van deze certificaten is gebaseerd op vertrouwen. In het certificaat geef je aan wie je bent als server. Als de client met bijvoorbeeld Microsoft wil communiceren en de reagerende server “zegt” ik heb een certificaat en daarop staat: “ik ben Microsoft” dan klopt dit en kunnen de gegevens veilig verzonden worden. Gelukkig is het niet zo dat iedereen kan zeggen “ik ben Microsoft” want dan was dit systeem volkomen nutteloos. In het certificaat staat namelijk de certificaat houder en een (digitale) handtekening, deze key is geverifieerd door een Certificate Authority, als deze Certificate Authority heeft geverifieerd dat jij Microsoft bent is je certificaat geldig. Gelukkig is het niet zo dat iedereen voor Certificate Authority kan spelen, er is een vast aantal Certificate Authorities en deze zijn betrouwbaar. Van deze Certificates Authorities is een lijst beschikbaar, daarmee kan gecheckt worden of een Certificate Authority werkelijk bestaat en te vertrouwen is. Deze lijst is standaard ingebouwd in de browser, het checken van de geldigheid van de uitgegeven certificaten is dus vanuit de client makkelijk te doen. Als een Certificate Authority de geheime keys bekend maakt aan anderen en daarmee zijn vertrouwen schaad wordt de Certificate Authority van de lijst afgehaald en is hij niet langer betrouwbaar en zijn de door hem uitgegeven certificaten ongeldig.

**Technische werking van het TLS-protocol**

**Tekst joris.**

**Hacks**

In dit hoofdstuk willen we ingaan op bekende hacks van het SSL/TLS en daarmee de betrouwbaarheid van https. We bekijken ook de veiligheid van de verschillende versies van SSL/TLS die nog in gebruik zijn.

**Beast attack**

In 2011 vond de zogenaamde **beast attack** plaats, dit is al weer even geleden en het is nogal een technisch verhaal om precies uit te leggen wat er gebeurde, het richt zich vooral op de ciphers binnen TLS 1.0 en SSL 3.0 die gebruik maken van een zogenaamde CBC modus. Dit zorgt ervoor dat TLS 1.0 en SSL 3.0 met de CBC modus niet meer veilig is. Vandaar dat het niet verstandig is om dit versie te gebruiken.

Meer over de impact van BEAST valt te lezen op:

<http://resources.infosecinstitute.com/ssl-attacks/>

<http://www.educatedguesswork.org/2011/09/security_impact_of_the_rizzodu.html>

**Crime**

## In 2012 hadden we te maken met crime (Compression Ratio Info-Leak Mass Exploitation

), dit was een aanval die zich richtte op compressie in TLS 1.0. Het gebruiken van TLS compressie is sindsdien af te raden.

Meer over crime valt te lezen op:

<http://resources.infosecinstitute.com/beast-vs-crime-attack/>

**Heartbleed**

Heartbleed is eigenlijk helemaal geen ingewikkelde en slimme hack, nee het is een simpele programmeerfout in de TLS implementatie van het heartbeat mechanisme in het opensource openSSL framework. De heartbeat berichten binnen het systeem worden niet goed beveiligd en zo kan er door een speciale request naar een beveiligde server te sturen veel informatie (tot 64kb) uit het geheime geheugen van de serveer terug gestuurd worden. Zoals private keys, TLS session keys, wachtwoorden en session ticket keys. Je hoeft niet perse te begrijpen wat al deze dingen zijn om te snappen hoe gevaarlijk deze programmeerfout is, om deze termen verder individueel te behandelen gaat namelijk buiten het bestek van deze website. Zoals je hopelijk wel begrijpt is het stelen van private keys het gevaarlijks omdat dan alle gegevens gedecrypt kunnen worden. De bug zit in open SSL 1.0.1 tot 1.0.1f. Dit lek werd ontdekt door medewerkers van Google en is naar buiten gebracht op 7 april 2014.

Meer over Heartbleed valt te lezen:

<http://heartbleed.com/>

**Poodle**

De POODLE aanval is ontdekt door medewerkers van Google en naar buiten gebracht op 14 november 2014, deze aanval was gericht op SSL 3.0 en betekende het definitieve einde van SSL 3.0, want SSL 3.0 wordt door POODLE in het algemeen beschouwd als onveilig. Deze POODLE aanval was niet gericht op de server kant zoals Heartbleed en Shellshock maar op de client kant, de browsers. Veel TLS clients (bijvoorbeeld browsers) hebben om te kunnen communiceren met verouderde servers de mogelijkheid om de TLS/SSL versie te downgraden naar de hoogst gesupporte versie van TLS/SSL op deze verouderde server. Hierdoor is het mogelijk door hackers de browser terug te schakelen naar een onveilige versie van SSL. Daarmee kan de eerder geencrypte data worden onthuld. Zo kunnen bijvoorbeeld je session cookies gestolen worden.

Meer over poodle is te lezen in:

<https://www.openssl.org/~bodo/ssl-poodle.pdf> of <http://blog.erratasec.com/2014/10/some-poodle-notes.html#.VPRvovmG-Ck>

**TLS is niet voldoende**

http://www.nist.gov/manuscript-publication-search.cfm?pub\_id=915295

**Toekomstverwachting**

Via https is kan informatie heel veilig verzonden worden, TLS is een goede beveiligings laag maar het hele systeem is gebaseerd op een stukje vertrouwen, als dit vertrouwen geschaad wordt kan het verkeer direct onveilig worden, vandaar dat het nodig is TLS voortdurend te ontwikkelen en te verbeteren helemaal door de verschillende aanvallen die er zijn, om zich zo te blijven wapenen en veilig verkeer te waarborgen. Maar TLS is niet genoeg, als het internet verkeer veilig is maar de rest van de we applicatie is niet veilig kan het verkeer als nog onderschept worden.

Wij denken dat het niet lang meer duurt voordat https de standaard wordt voor het verzenden van informatie over het internet. Vooral het verkeer waarbij er client-server relatie is en waarbij er dus gegevens worden verzonden tussen de client en server. Je moet hierbij denken aan inlog processen, e-commerce afreken systemen en dergelijke. Google gaat in Google Chrome namelijk http verkeer als onveilig beschouwen, er komt een icoontje bij google chrome die waarschuwt voor onveilige verbindingen die geen https hebben geïmplementeerd. Http wordt dan simpel weg een ongeaccepteerde manier voor het verzenden van informatie.

Bronnen:

1) <http://www.gemalto.com/identity/inspired/secure-ecommerce/ssl>

2) <https://www.evsslcertificate.com/ssl/ssl-history.html>

3) <http://www.ibm.com/developerworks/library/ws-ssl-security/>

4)

[https://tools.ietf.org/html/rfc6176](https://tools.ietf.org/html/rfc6176 5)

[5](https://tools.ietf.org/html/rfc6176 5)) <http://conferences.sigcomm.org/imc/2013/papers/imc257-durumericAemb.pdf>

<https://www.globalsign.com/en/ssl-information-center/what-are-certification-authorities-trust-hierarchies/>

<https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet>

<http://blog.erratasec.com/2014/10/some-poodle-notes.html#.VPRvovmG-Ck>

<https://www.openssl.org/~bodo/ssl-poodle.pdf>