

# Beveiliging van netwerken en computers

Bert De Saffel

Master in de Industriële Wetenschappen: Informatica Academiejaar 2018–2019

Gecompileerd op 7 november 2018

# Inhoudsopgave

<b>1</b>	<b>Basisconcepten</b>	<b>2</b>
<b>2</b>	<b>Netwerk en communicatiebeveiliging</b>	<b>3</b>
2.1	SSH . . . . .	3
2.1.1	Architectuur . . . . .	4
<b>3</b>	<b>Encryptiealgoritmen</b>	<b>7</b>
<b>4</b>	<b>Software- en systeembeveiliging</b>	<b>8</b>

## Hoofdstuk 1

# Basisconcepten

## Hoofdstuk 2

# Netwerk en communicatiebeveiliging

Voorlopig hebben we enkel de basisconcepten gezien zoals: symmetrische encryptie, asymmetrische encryptie, hashfuncties en message authentication codes. Dit hoofdstuk zal deze concepten toepassen om beveiligingsprotocollen te ontwikkelen voor netwerken.

### 2.1 SSH

SSH (Secure Shell) is een **applicatielaagprotocol**. Ondanks deze naamgeving bevat SSH ook een **transportlaagprotocol**, met als bedoeling een veilige connectie te maken tussen andere OSI applicatielaagprotocollen (bv HTTP) en de werkelijke OSI transportlaag. SSH kan draaien op zowel workstations als routers en switches. Routers en switches volgen de OSI laag namelijk niet strict, zodat zij ook een applicatielaag hebben. Vroeger werd het programma **telnet** gebruikt om remote configuratie toe te passen. Deze manier van werken is **onveilig** aangezien verkeer tussen de local host en remote host niet geëncrypteerd wordt, zodat deze door eender wie kunnen bekeken worden. SSH verhelpt dit probleem door de informatie te encrypteren. Vrijwel alle UNIX en Linux distributies komen met een versie van SSH geïnstalleerd. SSH laat onder andere toe om:

- een veilige remote verbinding op te zetten vanuit eender welke local host naar eender welke remote host. Het protocol maakt gebruik van erkende algoritme voor zowel encryptie (sleutels van ten minste 128 bits), data-integriteit, sleuteluitwisselingen en public key management. Tijdens het leggen van een connectie worden algoritmen tussen de local host en remote host afgesproken, op basis van welke ze al dan niet ondersteunen,
- TCP te tunnelen via een SSH connectie,
- bestanden te verplaatsen, gebruik makend van bijhorende protocols zoals SCP (Secure Copy) of SFTP (SSH File Transfer Protocol),
- zowel X-sessions als poorten te forwarden.

### 2.1.1 Architectuur

SSH kan opgesplitst worden in drie elementen: Transport Layer Protocol, User Authentication Protocol en het Connection Protocol.

#### Transport Layer Protocol

Dit protocol heeft onder andere de verantwoordelijkheid om: authenticeren van servers, sleuteluitwisselingen en perfect forward privacy implementeren. Perfect forward privacy wil zeggen dat, indien een sleutel gecompriemd wordt tijdens een sessie, deze geen invloed kan hebben op de beveiliging van vorige sessies. Dit protocol loopt uiteraard op de transportlaag, en in de meeste gevallen zal dit TCP zijn. Vooraleer de cliënt kan connecteren moet hij de **public host key** van de server bezitten. Hiervoor kunnen er twee modellen gebruikt worden (cfr. RFC 4521) waarbij:

1. de cliënt een lokale databank bevat die de mapping beschrijft van elke hostnaam naar de corresponderende public host key. Op deze manier is er geen centrale entiteit nodig, maar het beheer van deze databank kan, indien deze groot genoeg wordt, lastig zijn om te onderhouden.
2. de mapping bevestigd wordt door een **CA (Certification Authority)**. De cliënt kent in dit geval enkel de CA root key waarmee de geldigheid van elke host key kan nagaan die getekend zijn door deze CA. In dit geval moet de cliënt slechts één of enkele CA root keys bevatten.

Wanneer er een connectie kan gelegd worden van een cliënt tussen een server, is het eerste proces altijd het **onderhandelen van de algoritmen**. Deze stap zal algoritmen selecteren die compatibel zijn met zowel de cliënt als de server. Wanneer de cliënt een TCP connectie heeft met de server worden volgende pakketten verstuurd:

- **Identification string exchange.** Dit is een string dat zowel het protocolversie als de softwareversie van SSH bevat. Deze string wordt ten eerste van de cliënt naar de server verstuurd, waarop de server dan antwoord met zijn protocol en softwareversie. Indien hieruit blijkt dat de machines niet compatibel zijn met elkaar, wordt de connectie onderbroken (**SSH2 is bv niet compatibel met SSH**) en worden volgend stappen bijgevolg niet meer uitgevoerd.
- **Algorithm Negotiation.** In deze fase sturen zowel de server als de cliënt een SSH\_MSG\_KEXINIT bericht, die een lijst van alle ondersteunde algoritmen bevat, in volgorde van voorkeur. Elk type van algoritme zoals sleuteluitwisseling, encryptie, MAC algoritme en compressiealgoritme heeft zo zijn eigen lijst. Elk type moet dan ook een algoritme toegekend krijgen en wordt bepaald door het eerste algoritme dat de cliënt goed vindt dat ook beschikbaar is op de server.
- **Key Exchange.** Indien de vorige fase goed gelukt is, start nu de sleuteluitwisseling. Voor de sleuteluitwisseling worden er momenteel slechts twee versie van **Diffie-Helman** ondersteund (cfr. RFC 2409). Het einde van de sleuteluitwisseling wordt signaleerd door een SSH\_MSG\_NEWKEYS pakket, met als gevolg dat zowel de cliënt als de server de gegenereerde sleutels mag gebruiken.
- **Service Request.** De laatste stap, dat eigenlijk het begin is van een volgend proces, wordt signaleerd door een SSH\_MSG\_SERVICE\_REQUEST pakket. Dit pakket vraagt ofwel de start van het User Authentication Protocol of van het Connection Protocol. Alle verkeer

tussen server en cliënt wordt op dit moment getransporteerd als de payload van een SSH Transport Layer pakket, beveiligd met encryptie en een MAC.

Een SSH Transport Layer pakket heeft de volgende vorm, waarbij elementen met een  $\delta$ -symbool geëncrypteerd en geauthenticeerd zijn en elementen met een  $\Delta$ -symbool ook optioneel gecompresseerd kunnen zijn:

$\delta$  **Pakketlengte (4 bytes)**. De lengte van het pakket in bytes, zonder de lengte van de pakketlengte zelf en het MAC veld in beschouwing te nemen.

$\delta$  **Paddinglengte (1 byte)**. Dit is de lengte van het random padding veld.

$\Delta$  **Payload**. De eigenlijke informatie van het pakket.

$\delta$  **Random padding**. Dit veld dient om cryptanalyse moeilijk te maken. Kleine pakketten zijn op deze manier minder veel minder voorspelbaar en algoritmen, die vaak een kenmerkende vaste lengte hebben, worden ook moeilijker te achterhalen.

/ **MAC veld**. Dit veld wordt enkel gegenereerd indien dit zo in de onderhandeling besproken werd. Dit veld wordt berekend over het hele pakket en krijgt ook nog een bijhorend sequentienummer. Het sequentienummer start op 0, en wordt telkens met 1 geïncrementeerd voor elk pakket. Een aanvaller kan dit sequentienummer niet achterhalen aangezien een MAC een onomkeerbaar proces is.

Het **sleuteluitwisselingsproces** vraagt wat enige uitleg, er is echter nog niet nagegaan of deze sleutels op een **veilige** manier uitgewisseld worden. Op het moment van sleuteluitwisseling is er helemaal nog geen SSH connectie. Het uitwisselingsproces verloopt in twee fasen: eerst wordt er een gedeelde sleutel gegenereerd met Diffie-Helman, daarna wordt deze gedeelde sleutel gesigneerd met de publieke sleutel van de cliënt voor authenticatie. Sleutels kunnen ook heruitwisseld worden, hierbij gelden volgende regels:

- Mogelijk op elk moment, behalve tijdens het sleuteluitwisselingsproces.
- Kan aangevraagd worden door beide partijen.
- Sessie identificaties blijven ongewijzigd.
- Cryptografische algoritmen kunnen gewijzigd worden.
- Sessiesleutels worden vervangen.

Meestal worden sleutels vervangen na het behalen van een bepaalde quota zoals een tijdslimiet of het aantal totaal verstuurd bytes.

## User Authentication Protocol

Dit protocol specificeert **hoe** een cliënt zich moet authenticeren aan een server. Meerdere methoden zijn mogelijk waaronder de drie belangrijkste ervan: Public Key Authentication, Password Authentication en Host Based Authentication:

- **Public Key Authentication.** De implementatie van deze methode is afhankelijk van het gebruikte public-key algoritme. Een cliënt verstuurt berichten, dat gesigneerd is door de cliënt zijn private sleutel, naar de server, die de publieke sleutel van de cliënt bevat. De server gaat na of deze publieke sleutel nog geldig is en zoja, of dat de signatuur correct is.
- **Password Based Authentication.** De cliënt verstuurt zijn paswoord, dat geëncrypteerd wordt door het SSH Transport Layer Protocol naar de server, die nagaat of het paswoord correct is.
- **Host Based Authentication.** \_ToDo:

## Connection Protocol

\_ToDo: vanaf slide 26

## Hoofdstuk 3

# Encryptiealgoritmen

Encryptiemethoden worden al eeuwenlang gebruikt om informatie onleesbaar te maken voor partijen die deze informatie niet mogen achterhalen. Deze inleiding bespreekt de basismethoden om encryptie toe te passen.

Zogenaamde **Substitution ciphers** zijn de meest eenvoudigste vorm van encryptie. De originele versie vervangt elke letter van het alfabet met een voorgedefinieerde shift in positie. Een shift van 3 zal het originele alfabet (in kleine letters) afbeelden op het verschoven alfabet (in grote letters): Het

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

duidelijke nadeel is dat er slechts 25 verschillende encryptiemogelijkheden zijn. Een verbeterde versie mapt elke letter met een willekeurige andere (nog niet gebruikte) letter: waardoor er nu  $26! \approx 4 \cdot 10^{26}$

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	K	V	Q	F	I	B	J	W	P	E	S	C	X	H	T	M	Y	A	U	O	L	R	G	Z	N

mogelijkheden zijn. Deze manier is daarom niet meer secuur, aangezien de onderliggende frequentie van letters nog steeds dezelfde is. Een eenvoudige dencryptiemethode is om de relatieve frequenties van de ciphertekst te tellen, waardoor er statistisch kan achterhaald worden met welke gedecrypteerde letter elke geëncrypteerde letter overeenkomt. **ToDo: vanaf slide 21 C04**



## Hoofdstuk 4

# Software- en systeembeveiliging