Het bouwen van een veilige web server

Erik Meinders
Jos Visser

Open Solution Providers
Dalsteindreef 16
1112 XC Diemen
tel: 020-4950222
fax: 020-4950223
http://www.osp.nl

13 juli 2004

Geformatteerd (en hoogstwaarschijnlijk afgedrukt) op 13 juli 2004.

Inhoudsopgave

Module 1

Inleiding

1.1 Inleiding

Het bouwen van een veilige web server

Erik Meinders
Jos Visser
http://www.osp.nl

Notities

Welkom bij de speciale eendaagse minicursus over het opzetten van een veilige webserver. Deze dag wordt verzorgd door de Open Solution Providers.

1.2 Inhoud

Inhoud

De grondslagen van het web
Het opzetten van een web server
De beveiliging van een web server

Notities

Deze dag bestaat uit drie gedeelten. In het eerste gedeelte laten we de algemene grondslagen van het World Wide Web de revue passeren. Voor een aantal deelnemers zal deze informatie al grotendeels bekend zijn. Algemene termen als HTTP, HTML en CGI worden nog eens uitgediept en uitgelegd. Vervolgens gaan we in op het implementeren van een web server, de benodigde hard- en software en algemene beheerzaken zoals logging, accounting en tuning. De middagsessie gaat in zijn geheel over WWW beveiliging, zowel vanuit de client als vanuit de server bezien.

Voor het volgen van deze mini-cursus is basiskennis over TCP/IP en Internet nodig. In de bijlage vindt u een uitgebreide inleiding over TCP/IP.

Deel I

De grondslagen van het web

Module 2

Inleiding World Wide Web

2.1 Hoe het zo gekomen is...

Hoe het zo gekomen is...

Mijlpalen: 1965, 1989, 1991, 1993
Populaire applicaties: telnet, email, ftp, gopher, WAIS
CERN: Tim Berners Lee
HTTP en HTML standaarden
W3 Consortium
Mosaic
Netscape

Notities

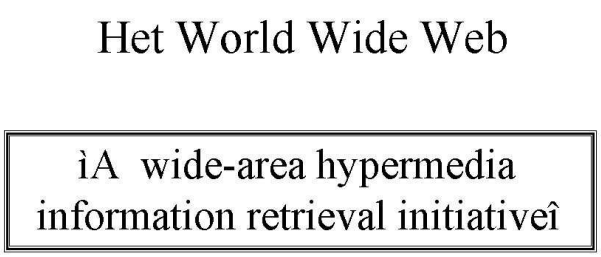
Het idee van een op hypertekst gebaseerde informatiedienst is ouder dan je wellicht zou denken. Al in 1945 werd een systeem beschreven (MEMEX) voor het opslaan en toegankelijk maken van informatie via associatieve indexen. In 1965 beschreef Ted Nelson het “Xanadu” systeem: een universum van onderling verbonden documenten voor het toegankelijk maken van informatie.

In 1989 begon de aan CERN verbonden Tim Berners Lee met de ontwikkeling van standaarden voor het electronisch publiceren van wetenschappelijke rapporten. In dergelijke rapporten zitten traditiegetrouw enorm veel verwijzingen naar andere rapporten. Het systeem van Tim moest het mogelijk maken van van het ene document naar het andere te springen zonder dat de gebruiker door had waar die documenten uithingen en hoe ze daar heetten. Deze standaarden, HTTP en HTML, werden, zoals in Internet gebruikelijk is, aan het grote publiek bekend gemaakt. De populariteit van dit “World Wide Web” nam met sprongen toe toen de programmeurs van het “National Centre for Supercomputing Application” (NCSA) een uiterst fraaie en gebruikersvriendelijke client voor het World Wide Web ontwikkelden en in het public domain stopten. Deze client (Mosaic) verwierf in “no time” een grote schare volgelingen. Het was ook eigenlijk het eerste gebruikersvriendelijke programma voor het gebruik van Internet.

Marc Andreesen, het brein achter Mosaic, richtte samen met enkele anderen het bedrijf Netscape Communication op. Netscape ontwikkelde in vervolg op Mosaic de bekende Netscape Navigator browser. De kracht en flexibiliteit van Netscape verleidde nog eens vele duizenden tot het inrichten van World Wide Web sites en het gebruik van het web. De hieruit voortvloeiende hausse op het gebied van het web heeft de populariteit van Internet tot ongekende hoogte opgestuwd. Voor een groot aantal gebruikers is het

Internet gelijk aan het World Wide Web!

2.2 Het World Wide Web



Notities

Wat is het World Wide Web?

Deze vraag is niet eenvoudig te beantwoorden. Het hedendaagse web is veel verschillende dingen voor veel verschillende mensen. De definitie van het W3 Consortium staat op de slide.

2.3 Het World Wide Web (again)

Het World Wide Web

Een gedistribueerde hypertext informatieservice

Een interactieve online transactieservice

Toepassingen:

- Electronische informatiediensten
- Web applicaties, bijvoorbeeld
  - Electronische winkels
  - Raadplegen databases

Notities

De twee belangrijkste functies van het web zijn:

1. Een gedistribueerde hypertext informatieservice
   De mogelijkheden van het World Wide Web zijn veelvuldig toegepast voor het inrichten van electronische informatiediensten. Voorbeelden hiervan zijn electronische catalogi, online handleidingen en electronische folderstands. Het gemak waarmee een WWW-site kan worden ingericht heeft ervoor gezorgd dat niet alleen grote professionele organisaties Web sites hebben ingericht. Kleine bedrijven, stichtingen en privé personen zijn ook grootscheeps overgegaan tot het opzetten van zogenaamde ‘home pages’. De hoeveelheid beschikbare informatie op het WWW is daardoor zeer gigantisch.

2. Een interactieve online transactieservice
   Met de standaard mogelijkheden van het World Wide Web kunnen op eenvoudige wijze simpele transactiemogelijkheden worden geïmplementeerd. (Complexe interactie is ook mogelijk, maar is minder simpel.) Op basis van deze faciliteiten zijn er al een groot aantal sites waar niet alleen informatie kan worden opgehaald, maar waarmee online interactief mee kan worden gecommuniceerd. Zo kunnen bezoekers bijvoorbeeld online een spel spelen, hun naam achterlaten (voor het per landpost opsturen van meer informatie) of een bestelling doen. Op basis van deze faciliteiten zijn ondertussen al een aantal electronische winkels gebouwd.

2.4 Populariteit van het WWW

Populariteit van het WWW $f\infty\infty\infty\infty$

Grafische user interfaces
Gebruikersvriendelijk
Platform onafhankelijk
Open standaard
Eenvoudig
- Voor de gebruiker
- Voor de implementator (vroeger)
- Voor de webmaster (relatief)

Notities

Het World Wide Web is in zeer korte tijd ongekend populair geworden en heeft in zijn kielzog het Internet meegesleurd in de vaart der volkeren. Ondanks het feit dat het WWW slechts één van de vele mogelijke Internet applicaties is (en een zeer recent ontwikkelde) denken een groot aantal gebruikers dat Internet en het World Wide Web synoniemen van elkaar zijn.

De populariteit van het World Wide Web valt uit de volgende factoren te verklaren:

- Grafische gebruikers interface
  Het WWW is één van de weinige applicaties met een fraai grafisch gebruikersinterface. Oudere Internet applicaties waren ook al eens voorzien van een grafisch interface, maar in dit geval blijft het altijd een oude tekst geörienteerde applicatie die van een grafisch jasje is voorzien.

- Gebruikersvriendelijk
  De World Wide Web client software is zeer gebruikersvriendelijk van aard. Dit is een grote tegenstelling met de traditionele Internet client programmatuur die eigenlijk alleen door automatiseerders kan worden gebruikt. Het WWW is eigenlijk de eerste eindgebruikersgerichte applicatie in het Internet.

- Platformonafhankelijk
  Het World Wide Web is platformonafhankelijk. Het maakt absoluut niet uit met wat voor soort computer of besturingssysteem de client en de server zijn uitgerust. In het heterogene Internet waar een grote verzameling hard- en software rondtolt is dit een groot voordeel. De toegankelijkheid van het World Wide Web

is hierdoor zeer goed. Het maakt niet uit wat voor soort computer of OS je hebt, er is altijd wel een client of server software pakket te vinden.

- Open standaard
  Het World Wide Web is gebaseerd op open en goed gedefinieerde en toegankelijke standaarden. Hierdoor kan er vrij eenvoudig nieuwe client en server programmatuur worden ontwikkeld.

- Eenvoudig
  Het World Wide Web is eenvoudig. Gebruikers die al enige ervaring hebben met moderne grafische applicaties behoeven geen nadere toelichting bij het gebruik van hun WWW client. Ook het opzetten van een WWW site is tamelijk eenvoudig. De hiervoor benodigde software is goed verkrijgbaar en goedkoop (meestal public domain). Een groot aantal Internet providers biedt haar klanten de mogelijkheid om één of meer pagina's op het Web te publiceren. Hierbij heeft de provider de benodigde infrastructuur geregeld en hoeven de gebruikers alleen nog maar de pagina's te bedenken. De programmeertaal waarin deze pagina's moeten worden ontwikkeld is ook redelijk simpel.

  Het schrijven van WWW client- en server software was met name vroeger ook niet al te moeilijk. De toenemende concurrentie tussen Microsoft en Netscape heeft er echter voor gezorgd dat de moderne WWW clients ongekend krachtig zijn en vol met features zitten. Het ontwikkelen van een nieuwe client is hierdoor een niet-triviale aangelegenheid geworden.

2.5 Organisatie van het web

Organisatie van het web

W3 Consortium
Internet Architecture Board (RFCís)
Leveranciers:

Microsoft	VBScript, ActiveX
Sun	Java
Netscape	JavaScript, SSL
TIS	S-HTTP

Notities

De standaardisering van het web is momenteel in handen van drie groepen (instanties):

1. Het W3 consortium
   Dit is een groep instellingen, personen en bedrijven die onder aanvoering van het MIT het gebruik en de standaardisering van het World Wide Web willen bevorderen. Zij bedenken nieuwe standaarden en voorzien in referentie implementaties van die standaarden.

2. Het Internet Architecture Board
   Het IAB houdt zich bezig met de standaardisering van Internet technologie door middel van RFC's. Bepaalde onderdelen van het WWW zijn inmiddels al in een RFC vastgelegd.

3. Leveranciers
   Producenten van Internet software zijn momenteel erg actief in het verzinnen van nieuwe uitbreidingen en het implementeren van die uitbreidingen in hun produkten. Leveranciers als Sun, Netscape en Microsoft komen om de haverklap met verbeteringen die ze vervolgens ook weer ter beschikking stellen aan hun concurrenten in de hoop een Internet brede de facto standaard neer te zetten. De ontwikkelingen volgen elkaar momenteel zo snel op dat de traditionele standaardiserings trajecten (IAB/RFC) vaak niet de tijd hebben de nieuwe technieken formeel te standaardiseren.

2.6 Hypermedia

HyperMedia

- Hypertext
  - Niet lineair
  - Zelfstandige teksteenheden
  - Teksteenheden onderling verbonden via *links*
- Hypermedia
  - Generalisatie van hypertext
  - Documenten met niet-tekstuele componenten
    - Plaatjes, geluid, invulvelden, video

Notities

De grondgedachte van het World Wide Web is gebaseerd op de begrippen *HyperMedia* en *HyperText*. Hypertext documenten bestaan uit onderling verbonden teksteenheden. Deze verbindingen (links) kunnen door gebruikers worden gebruikt om van de ene teksteenheid naar de andere te komen. Een normaal boek of artikel is *lineair*, het moet van voor naar achter worden gelezen. Hypertext is niet lineair. Iedere teksteenheid is een op zichzelf staande eenheid. Via de verbindingen kan van een teksteenheid naar een veelheid van andere teksteenheden worden gesprongen. Zo kunnen begrippen die niet geheel duidelijk zijn onmiddelijk worden verklaard door een verbinding op te nemen naar een teksteenheid die het begrip in kwestie verder uitdiept. Gebruikers lezen hypertext niet in de gebruikelijke zin des woords, ze navigeren door een universum van documenten (ook wel “docuverse” genoemd). Hierdoor kan de gebruiker de informatie tot zich nemen in de volgorde en op het detailniveau die aansluiten bij de kennis en ervaring van de gebruiker.

Het begrip “hypermedia” wordt in de literatuur gedefinieerd als een generalisatie van hypertext. In hypermedia documenten is niet alleen tekst opgenomen maar ook niet-tekstuele elementen zoals plaatjes, geluid en video. In de hedendaagse terminologie worden hypertext en hypermedia door elkaar gebruikt.

2.7 Hyper...

Hyper

Node, document

(Hyper)link

Logische verbinding tussen twee documenten

(Hyper)web

Verzameling links tussen hypermedia document

Hyperdocument

Logische verzameling documenten en links

Notities

Een “docuverse” van hypermedia documenten kan door de onderlinge verbindingen worden gezien als een netwerk, of een web. De knooppunten in dit web zijn de documenten, ook wel “nodes” genoemd. De verbindingen tussen de nodes zijn de hyperlinks.

Module 3

De structuur van het web

3.1 WWW Componenten

WWW Componenten

Client
- Browser

Server(s)
- HTTP, FTP, Telnet, Gopher, ...

Internet
- Communicatiemechanisme

Notities

Het World Wide Web is een client/server applicatie.

De gebruikers van het web draaien een WWW client applicatie (een zogenaamde browser). Deze browsers maken over het Internet contact met WWW server applicaties voor het ophalen van documenten en het opsturen van informatie. WWW browsers kunnen informatie ophalen van een groot aantal verschillende server applicaties. Naast de WWW-specifieke HTTP servers kunnen browsers ook informatie ophalen van FTP en Gopher servers. Hierdoor kunnen de gebruikersvriendelijke browsers ook worden gebruikt als user interface voor traditionele Internet applicaties. Browsers kunnen zelfs worden geïnstrueerd om telnet sessies met remote servers op te starten.

WWW Servers draaien server applicaties die op verzoek documenten en bestanden teruggeven aan WWW browsers. De server applicatie heeft meestal niet door of hij door een WWW client of door een andere client wordt benaderd. Het is de taak van de WWW client om het applicatieprotocol van de WWW server applicatie te praten.

3.2 WWW Transacties

WWW Transacties

Request/Response transacties

Retrieval

Browser haalt een document op bij een server

Protocollen: HTTP, FTP, Gopher

Reply

Browser stuurt informatie op naar een server

Invulformulieren

HTTP

Notities

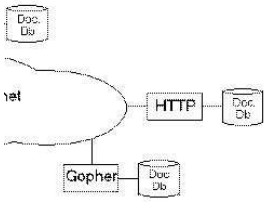
Het World Wide Web is een transactie geörienteerd medium. De client zet een verzoek uit bij een WWW server applicatie (*request*). De server beantwoord dit verzoek met het gevraagde document of met een foutmelding. Deze *response* wordt door de WWW client verwerkt. Meestal komt deze verwerking neer op het tonen van het opgehaalde document. De gebruiker kiest vervolgens één van de hyperlinks in het document, waarop de browser het volgende document ophaalt en zo voorts.

In principe kan een WWW client twee soorten transacties initiëren:

- Retrieval
  Met een “retrieval” transactie worden documenten van WWW servers opgevraagd. De client dient exact aan te geven waar het document huist, hoe het daar heet en met welk protocol het kan worden opgehaald. Het opgehaalde document wordt door de browser getoond.

- Submit
  Indien een document invulvelden bevat kan een gebruiker deze invullen en de ingevulde waarden terugsturen naar een WWW server. De WWW server verwerkt de opgestuurde waarden en produceert op basis daarvan weer een nieuw document. Met dit mechanisme kunnen onder andere electronische winkels en dergelijke in elkaar worden geknutseld. Momenteel ondersteunen alleen HTTP servers het opsturen van informatie. Gopher en FTP servers beperken zich tot het ophoesten van documenten op verzoek.

3.3 World Wide Web



Notities

Één van de kernmerkende eigenschappen van WWW is dat de gebruiker totaal het overzicht verliest van waar welke informatie vandaan komt. Dit komt omdat de verbindingen tussen hypermedia documenten vrij eenvoudig de grenzen van een server kunnen overschrijden. In het “adres” van een document (een zogenaamde URL) komt namelijk onder andere de hostnaam voor van de server waar een document thuishoort. De programmeurs van de WWW documenten kunnen bij het bouwen van de hypermedia pagina’s verbindingen opnemen naar ieder document op Internet. Dit gebeurt dan ook veelvuldig. Daarnaast zijn er gespecialiseerde “zoek-en-index” servers die er hun taak van maken om verwijzingen naar ieder WWW document op de planeet op te nemen. Hierdoor worden alle WWW servers van de wereld aan elkaar gekoppeld tot een wereldwijd web.

3.4 WWW Client

WWW Client

Browser
Gebruikersinterface
Haalt documenten op van servers
Geeft documenten weer
Invullen van documenten met invulvelden
Opsturen ingevulde formulieren

Notities

Iedere WWW eindgebruiker moet in het bezit zijn van een WWW client applicatie, de zogenaamde “browsers”. De eigenlijke taak van dergelijke browsers is eenvoudig: ze halen documenten op van WWW servers, geven deze weer en laten gebruikers met deze documenten interacteren (kiezen van hyperlinks, invullen van invoervelden, opsturen van een ingevuld formulier). De allereerste browsers waren inderdaad tamelijk eenvoudige applicaties. Uit de begintijd van Internet komt de opmerking: *“Every bozo can write a browser, a guy in our office wrote one in a week”*.

Vandaag aan de dag is dat zeker niet meer het geval. De concurrentie tussen grote software leveranciers (met name Microsoft en Netscape) heeft ervoor gezorgd dat browsers grote, complexe en featurevolle applicaties zijn geworden. De hedendaagse browser is groter dan de kernel van een operating system!

3.5 Voorbeelden van browsers

Voorbeelden van browsers

Netscape Navigator
Microsoft Internet Explorer
Mosaic
Cello
Lynx
CERN Line Mode Browser

Notities

Op de slide staan voorbeelden van bekende (en minder bekende) browsers.

3.6 Eigenschappen van browsers

Eigenschappen van browsers

Verschillende mogelijkheden

B.v. grafische mogelijkheden, geluid

Integratie met andere client applicaties (mail)

Active Content

Ondersteunde documenttypen

Verschillende weergave van hetzelfde document

Lokale *cache*

Notities

In de diverse standaarden is vrij goed vastgelegd waaraan browsers moeten voldoen. De implementatoren hebben echter behoorlijk wat vrijheid. Browsers verschillen met name in hun mogelijkheden voor het weergeven van niet-tekstuele elementen (plaatjes, video, geluid), ondersteuning van niet-standaard documenttypen (PostScript, Adobe Acrobat) en de wijze waarop ze zogenaamde “Active Content” ondersteunen. Het is absoluut niet voorspelbaar met wat voor soort browsers de gebruikers in Internet werken. Hierdoor moet de “webmaster” van een site goed nadenken voordat hij/zij features gebruikt die niet door alle browsers worden ondersteund. Er is niets zo vervelend als een web site die niet met de door jouw gebruikte browser kan worden bekeken. Vrij kenmerkend is dat verschillende browsers hetzelfde document op totaal verschillende wijze kunnen weergeven. Dit wordt niet veroorzaakt door verschillende interpretatie van de standaarden maar door een (toegestande) verschillende wijze van implementeren van de standaard. Veel webmasters hebben een slecht begrip over wat de standaarden op het gebied van WWW documentopmaak precies inhouden. Om mooiere documenten te maken als standaard mogelijk is worden alle registers opgetrokken en wordt er nogal eens gebruik gemaakt van *undocumented features*.

De browsers halen documenten op met een TCP verbinding (meestal door het Internet). Dit kan (met name bij grote documenten) een grote vertraging met zich meebrengen. Om deze reden leggen veel browsers een lokale cache van recent opgehaalde documenten aan. Zodra een gebruiker een document ophaalt kijkt de browser eerst in zijn lokale cache. Is het document daar nog/al aanwezig dan krijgt de gebruiker deze lokaal opgeslagen kopie te zien. Modernere browsers kunnen eerst nog een controle over Internet doen bij de WWW server om te kijken of het document intussen op de server verouderd is.

3.7 MIME

MIME9u9◊

Multipurpose Internet Mail Extensions
RFCís 1521 en 1522
Insluiten non-ASCII data in Internet email:
- Plaatjes
- Geluid
- non-ASCII karakter sets (b.v. ISO 8859-1)

Ook toegepast in WWW

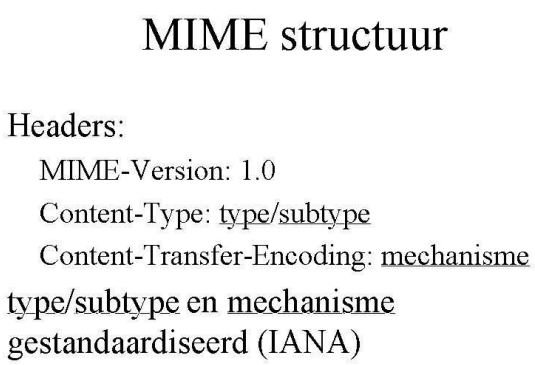
Notities

Om verschillende documenttypen te coderen wordt in het World Wide Web gebruik gemaakt van de MIME-standaard. MIME (Multipurpose Internet Mail Extensions) is ontwikkeld om andere dan de originele RFC 822 tekstberichten per electronische post te kunnen versturen.

Een behoorlijke beperking van de traditionele RFC 822 berichtenstandaard was dat er in principe alleen maar ASCII (7 bit) teksten in de body van het bericht konden worden meegestuurd. Moderne hard- en software hebben het mogelijk gemaakt om zonder problemen ook andere dan tekstuele data te manipuleren (multimedia). Het is dan natuurlijk gewenst om deze niet ASCII data ook via Internet email of het World Wide Web te kunnen versturen.

Deze uitbreidingen staan bekend onder de naam MIME: Multipurpose Internet Mail Extensions. MIME is gestandaardiseerd in RFC 1521 en 1522. Nieuwe MIME toepassingen (MIME types) worden in eigen RFC's gestandaardiseerd. MIME maakt het mogelijk om in de body van een email bericht allerlei non-ASCII data in te sluiten. De MIME standaard is zodanig opgesteld dat MIME berichten via standaard SMTP kunnen worden verstuurd. De MIME standaard is een berichtenstandaard en vereist dus geen uitbreidingen aan het transportprotocol.

3.8 MIME structuur



Notities

MIME introduceert een aantal nieuwe RFC 822 (en HTTP) headers. De aanwezigheid van deze headers vertelt het mail programma of de browser dat er sprake is van een MIME bericht. De ‘MIME-Version’ header is hiervoor de belangrijkste aanwijzing; deze geeft aan volgens welke versie van de MIME standaard het bericht is opgemaakt. Een MIME bericht bevat ook een ‘Content-Type’ header. Deze header geeft aan uit wat voor soort data de body bestaat. Een compleet MIME type bestaat uit een type, een subtype en optioneel een aantal parameters voor het type. De browser of het mail programma interpreteert het type en geeft de data op en geschikte wijze weer.

Aangezien MIME berichten met standaard email en HTTP protocollen moeten kunnen worden verscheept moet de body op een bepaalde manier worden bewerkt voordat het kan worden verstuurd. De ‘Content-Transfer-Encoding’ geeft aan met welk mechanisme de data is omgevormd tot transporteerbare tekst. Het ontvangende programma (mail programma of browser) dient deze bewerking om te keren voordat de orginele (non-ASCII) data weer bruikbaar is.

MIME types en ‘transfer encodings’ zijn Internet breed gestandaard bij de Internet Assigned Numbers Authority (IANA).

3.9 MIME header voorbeelden

MIME header voorbeelden

Content-Type:
- text/plain; charset=charset
- audio/wav
- text/html
- image/eps
- multipart/mixed; boundary=aboundarystring

Content-Transfer-Encoding:
- base64
- 8bit

Notities

Op de slide staan een aantal voorbeelden van MIME types en transport coderingen. Een MIME compatible browser interpreteert deze en kan dus de ingesloten data op een van toepassing zijnde manier aan de gebruiker tonen. Indien een bepaald MIME type niet kan worden ondersteund (bijvoorbeeld door het ontbreken van bepaalde hardware mogelijkheden zoals graphics, geluid of video) biedt de browser de mogelijkheid om de data op te slaan in een disk bestand.

Het bekendste MIME transportmechanisme is ‘base64’. Dit is een methode voor het (de)coderen van binaire data waarbij voor iedere 6 bits uit de binaire data een ASCII karakter in de reeks ‘A-Za-z0-9+-’ wordt gesubstitueerd. De base64 methode wordt ook gebruikt door andere Internet standaarden voor het coderen van binaire data.

3.10 MIME aandachtspunten

MIME aandachtspunten

Mail agent/Browser moet MIME begrijpen:
Benodigde bandbreedte
Beveiliging
Message/External-Body; access=ftp
Application/subtype (krachtige interpreters)

Notities

Om van MIME gebruik te kunnen maken is een MIME compatible programma nodig. Tegenwoordig ondersteunen vrijwel alle mail programma's en WWW browsers MIME.

MIME maakt het mogelijk om grote hoeveelheden data gemakkelijk in te sluiten te verzenden. De hiervoor benodigde Internet bandbreedte kan aanzienlijk oplopen. Denk hierbij niet alleen aan netwerkcapaciteit maar ook aan de benodigde disk ruimte op proxies en firewalls.

Bepaalde MIME types zijn gevoelig voor misbruik. Zo is er een MIME type 'message/external-body' wat aangeeft dat de eigenlijke body van het bericht apart door morden opgehaald. De parameters van dit type vertellen met welke methode de body moet worden opgehaald (bijvoorbeeld ftp). Als de browser dit automatisch doet bestaat de kans dat een snoodaard een MIME bericht verstuurt met daarin de opdracht om automatisch een bestand op te halen en op de hard disk op te slaan. Dit is natuurlijk een prachtige manier om virussen te verspreiden. Andere MIME types worden door zeer krachtige interpreters verwerkt. Deze staan soms het opnemen van commando's en macro aanroepen in de data toe (Postscript, Word documenten).

3.11 Documenttypen

Documenttypen

Afgeleid van MIME
Tekst
text/plain
HTML
text/html
Plaatjes
image/gif, image/jpeg, ...
Overige documenttypen

Notities

De World Wide Web browsers kunnen dus verschillende typen documenten tonen. De wijze waarop een document wordt getoond is grotendeels afhankelijk van het documenttype. Bekende WWW documenttypen zijn:

- text/plain
  Gewone ASCII tekst, wordt eenvoudigweg getoond.
- text/html
  Een document in de WWW documentopmaaktaal HTML. De browser interpreteert het document en toont het in opgemaakte vorm.
- image/gif, image/jpeg
  Plaatjes in een of andere codering. De browser toont het plaatje.

Andere bekende documenttypen zijn audio/wav en video/avi. Indien een browser een bepaald documenttype totaal niet begrijpt geeft hij een foutmelding op biedt hij aan het document op te slaan op de hard disk van de gebruiker.

3.12 Filetypes

Notities

Om een document correct af te beelden moet een browser natuurlijk wel in staat zijn om te bepalen wat het MIME documenttype van een document is. De wijze waarop hij dit doet is afhankelijk van de manier waarop het document is opgehaald:

- FTP, Gopher
  Bij bestanden die via FTP of Gopher worden opgehaald is de bestandsnaam bepalend voor het documenttype. De browser gebruikt de bestandsnaam als sleutel in een tabel van documenttypen. In deze tabel staat bijvoorbeeld vermeld dat bestanden waarvan de naam voldoet aan het patroon "*.doc" van het type application/msword zijn.
- HTTP
  Indien de documenten met het HTTP protocol worden opgehaald stuurt de HTTP server in het antwoord het MIME type mee. Onderdeel van het antwoord is de zogenaamde "Content-Type" header welke het MIME type van het meegestuurde document bevat.

3.13 WWW Server

WWW Server

Stuurt op verzoek een document (bestand) op.
Anonymous FTP server
Gopher server
HTTP server

Notities

Een WWW server heeft als belangrijkste taak het op verzoek opsturen van documenten. Op het moment dat het World Wide Web werd ontwikkeld werd er in Internet al volop gebruik gemaakt van Gopher en FTP, beiden applicaties voor het ophalen van documenten van remote servers. Het leek de bedenkers van het web dan ook een goed idee om deze server applicaties in het web te integreren. Een web browser kan dus documenten van zowel Gopher servers als anonymous FTP servers ophalen. Naast deze twee bestaande server applicaties is er ook een WWW specifiek protocol bedacht: HTTP. HTTP server applicaties zijn bij uitstek geschikt voor het aanleveren van documenten aan WWW browsers. Bepaalde WWW applicaties (zoals invoerformulieren) zijn alleen mogelijk met het HTTP protocol.

3.14 URL

URL

Uniform Resource Locator
ìAdresî van een document in het WWW
Worden geinterpreteerd door de browser
protocol://hostname[*:port*]*/URI*

Protocol:	ftp, http, https, gopher, mailto
Hostname:	Internet hostname (FQDN)
Port:	TCP poortnummer
URI:	Identificatie document binnen de server

Notities

In een wereldwijd systeem als het World Wide Web moet er natuurlijk een wereldwijd unieke adressering voor documenten zijn. Via deze adressering moet ieder document op de aardbol uniek te identificeren zijn. Binnen het WWW wordt gebruik gemaakt van URL's: *Uniform Resource Locator*. Een URL is een unieke aanduiding van een document en beschrijft:

- Het protocol waarmee het document moet worden opgehaald.
- De server waarvandaan het document moet worden opgehaald.

Optioneel De TCP poort waarachter die applicatie huist.
Indien deze niet wordt meegegeven neemt de browser aan dat de server applicatie achter de voor die applicatie gebruikelijke (well known) poort hangt.

Optioneel De naam van het document binnen de server (URI) Indien deze niet wordt meegegeven neemt de server een default document (meestal 'index.html').

Daar hostnamen in het Internet uniek zijn, en URI's binnen een server uniek moeten zijn is een URL een unieke documentadressering.

3.15 URI

URI

Uniform Resource Identifier

Formaat:

Padnaam	(bestandsverwijzing)
Padnaam#sectie	(intra document verwijzing)
Padnaam?data	(variabelen)

Notities

De benaming van een document binnen een server noemen we een *Uniform Resouce Identifier*, URI. De browser gebruikt het protocol, host en poortnummer gedeelte van een URL om te bepalen met welke server hij contact op moet nemen en hoe hij met deze server moet praten. Is de verbinding eenmaal tot stand gekomen dan gebruikt de browser de protocolspecifieke instructies om het document aangeduid door de URI op te halen. Een WWW server weet dus niet met welke URL hij is aangeduid, en waar die URL vandaan kwam.

URI's kunnen worden gezien als padnamen van bestanden op de WWW server. Het is echter niet gezegd dat de bestanden van de disk af komen (meestal wel). Het is ook goed mogelijk dat een WWW server wordt aangedreven door een database van documenten. In dat geval is de URI een andersoortige identificatie van het op te halen document. De browser kent meestal geen betekenis toe aan de URI, het is hoofdzakelijk een identificatie voor de WWW server.

Via een speciale syntax kan met een URI meteen worden doorverwezen naar een sectie in een document (URI#sectie). Binnen het document moet dan op één of andere manier (via HTML instructies) zijn aangegeven waar de sectie begint. In dit geval wordt de URI wel (gedeeltelijk) ook door de browser geïnterpreteerd. Ook kan een URI worden gebruikt om variabelen mee te geven aan de WWW server. Via de syntax URI?var1=a&var2=b kan een lijst van variabelen worden doorgegeven. Deze mogelijkheid wordt alleen toegepast in het HTTP protocol voor het doorgeven van de ingevulde velden op een invoerformulier.

3.16 URL voorbeelden

URL voorbeelden

http://www.fbi.gov
gopher://gopher.osp.nl
ftp://ftp.uu.net/pub/rfc/rfc822.txt.gz
http://localhost:8080/aap
http://www.osp.nl/triviant
https://www.shop.com/order#howto
http://www.osp.nl/cgi-bin/viewrfc?n=822

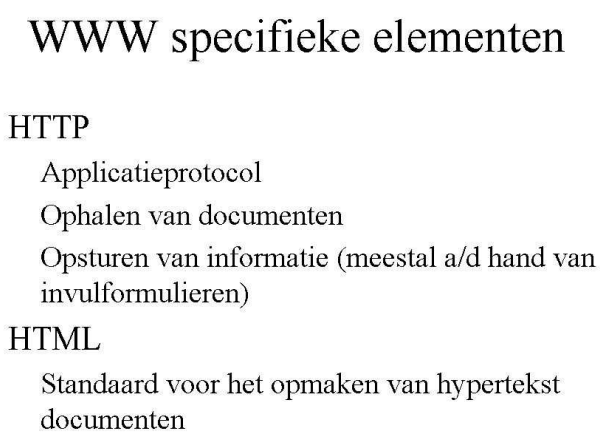
Notities

Op de slide staan een aantal voorbeelden van bekende en minder bekende URL's.

Module 4

HTTP en HTML

4.1 WWW specifieke elementen



Notities

Naast ondersteuning voor bekende applicaties en documenttypen kent het World Wide Web twee “eigen” elementen: HTTP en HTML. HTTP is een lichtgewicht toestandsloos protocol voor het ophalen van documenten en het opsturen van informatie. HTML is *de* standaard voor het opmaken van hypermedia documenten in het World Wide Web. Alle World Wide Web gebruikers en webmasters maken gebruik van deze twee elementen.

4.2 HTML

HTML¸◊9◊

HyperText Markup Language

Taal voor het logisch beschrijven van een document

De HTML interpreter (browser) bepaalt hoe het document er fysiek uit komt te zien

HTML documenten zijn tekstbestanden

HTML *tags* bevatten de metainformatie

Notities

Één van de grondslagen van het World Wide Web is het toegankelijk maken van informatie op basis van hypermedia documenten. Deze documenten kunnen naast tekst ook andere elementen bevatten zoals plaatjes en verbindingen met andere documenten. Dit soort elementen moeten dus op één of andere manier in de documenten kunnen worden "gehangen". De "webfathers" hebben er voor gekozen om de WWW hypermedia documenten op te stellen in een ASCII gebaseerde opmaaktaal: HTML, de HyperText Markup Language. Met behulp van deze opmaaktaal wordt de inhoud van een document logisch gespecificeerd. De HTML-programmeur geeft met HTML-opdrachten (zogenaamde *tags*) in de tekst weer hoe de structuur van het document is. Niet-tekstuele elementen en metainformatie (zoals verwijzingen naar andere documenten) worden met HTML-tags gespecificeerd. HTML lijkt wat dat betreft veel op andere opmaaktalen zoals troff (UNIX), DCF (VM en MVS) en TEX.

De browser die een HTML-document moet weergeven interpreteert de tags en doet zijn best om het document af te beelden. Hierbij is het aan de browser om te bepalen hoe bepaalde elementen worden weergegeven. Verschillende browsers geven bijvoorbeeld hoofdstuktitels en verwijzingen naar andere documenten verschillend weer. Hierdoor kan hetzelfde document er in twee browsers tamelijk verschillend uitzien. Zo mag een browser ook zelf bepalen wat hij doet met elementen die niet kunnen worden afgebeeld (door hardware/software beperkingen).

4.3 HTML voorbeeld

HTML voorbeeld

```
<html>
<head>
<title>DutchWorks Internet Seminar</title>
</head>
<body>
<h1>Voorbeeld document</h1>
Dit is een voorbeeld document. De items tussen <> zijn de
HTML <i>tags</i>. Druk <a href="nextdoc.html">hier</a>
voor het volgende document.
</body>
</html>
```

Notities

Op de slide ziet u een voorbeeld van een HTML-document. De elementen tussen de 'kleiner-dan' en 'groter-dan' tekens zijn de HTML-tags die vertellen hoe het document moet worden opgemaakt. Voor een volledige beschrijving van de HTML taal verwijs ik volgaarne naar één van de vele goede boeken op dit gebied.

4.4 HTML voorbeeld in IE

Notities

Op deze slide ziet u hoe het voorbeeld van de vorige slide eruit ziet indien het wordt getoond door de Microsoft Internet Explorer. Zoals u ziet interpreteert IE de HTML en geeft het document opgemaakt weer. Met een optie van de browser kunnen we meestal de originele HTML source weer boven water toveren.

4.5 HTML voorbeeld in lynx

Notities

Ziehier hetzelfde document, maar nu afgebeeld door de lynx browser (universiteit van Kansas). Lynx is een browser voor ASCII beeldschermen. Hij kan dan ook een aantal tags niet of nauwelijks weergeven. Toch doet ook lynx zijn best om het document correct af te beelden. Documenten die alleen maar grafische elementen bevatten kunnen met lynx dus niet worden bekeken!

4.6 HTML faciliteiten

HTML faciliteiten

Algemene opmaak

Hoofdstuk, paragraaf, lijsten, vet, *schuin*

Invoegen van plaatjes (<img> tag)

Invulvelden (tekst, knopje, *listbox*)

Gesplitste documenten (*frames*)

Notities

In HTML zitten allerlei faciliteiten voor het opmaken van documenten. Hieronder vallen tags voor logische opmaak (hoofdstuktitels, lijsten), fysieke opmaak (vet, schuin, gecentreerd) en invoerformulieren (knopjes, invulvelden). De meest recente uitbreidingen aan HTML maken het mogelijk om gesplitste documenten met subdocumenten, ook wel *frames* genoemd, te maken.

4.7 HTML aandachtspunten

HTML aandachtspunten

Daadwerkelijke presentatie afhankelijk van de browser

Verschillende HTML versies:

- HTML 1.0
- HTML +
- HTML 2.0
- HTML 3.0

Browser specifieke uitbreidingen

Notities

Wat erg belangrijk is bij het programmeren van HTML is dat browsers een grote vrijheid hebben bij het bepalen van hoe het document er uiteindelijk uit komt te zien. Het is dan ook aan te bevelen om een pagina met verschillende browsers zelf te bekijken alvorens hem in produktie te brengen. Verder zijn er verschillende versies van HTML geweest, en nog steeds in omloop. Dit betekent dat de allernieuwste HTML features niet door alle momenteel gebruikte browsers worden begrepen. Niet alle gebruikers upgraden continu naar de nieuwste versies van de browsers (bijvoorbeeld omdat ze nog vast zitten aan Windows 3.1).

Tot overmaat van ramp hebben implementeren sommige browsers hun eigen uitbreidingen aan HTML. Dit betekent dat die browsers bepaalde niet-standaard HTML tags begrijpen. Een document dat van die tags gebruik maakt kan niet door andere browsers worden geïnterpreteerd.

4.8 HTML, pro en contra

HTML, pro en contra

Pro:

Eenvoudig

Contra:

Veel verschillende versies in omloop

Zwakke *markup language*

Mengsel van logische en fysieke opmaak (b.v. de <h1> versus de <i> en <b> tags)

Geen zoek of index tags

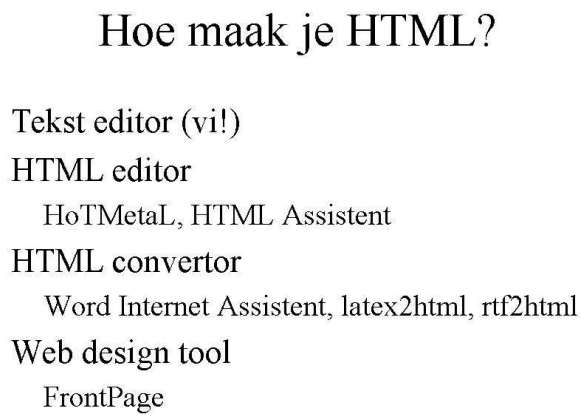
Notities

Het grote argument voor HTML is de ongekende simpelheid. Populair gesproken kan iedere dwaas met enige automatiseringskennis een HTML pagina in elkaar draaien. De eenvoud van HTML heeft er ook zorg voor gedragen dat er al snel na een groot aantal browsers was die HTML documenten goed konden weergeven.

Er zitten echter ook een groot aantal nadelen aan HTML: er zijn veel verschillende versies in omloop, het is een zwakke opmaaktaal met weinig mogelijkheden en er ontbreken een aantal features die het zoeken en indexeren van HTML documenten zouden vergemakkelijken. Minder belangrijk (maar daarom niet minder waar) is dat HTML geen “schone” opmaaktaal is; er zitten zowel tags in met een logische betekenis (bijvoorbeeld de hoofdstuktitel tag) als tags met instructies voor de fysieke opmaak (de *bold* tag).

Ondanks alle problemen is HTML zeer wijd verspreid. Het is een van de belangrijkste bouwstenen van het hedendaagse WWW en kan mede om die reden niet snel worden vervangen. Betere opmaaktalen zijn vaak dusdanig gecompliceerd dat ze niet snel kunnen worden getoond, of staan niet het niveau van controle toe dat webmasters over de opmaak van de pagina’s willen hebben.

4.9 Hoe maak je HTML?



Notities

Op zijn eenvoudigste manier kunnen HTML documenten worden aangemaakt met een willekeurige tekst editor. HTML documenten zijn immers ASCII tekstbestanden met daarin diverse ASCII opmaakcommando's. Een probleem met deze aanpak is dat de HTML-programmeurs de syntax van de verschillende HTML tags uit zijn/haar hoofd moet kennen. Verder bevatten de hedendaagse HTML documenten allerlei niet-tekstuele componenten (meestal plaatjes) die in geen geval met een tekst editor kunnen worden aangemaakt. Om die reden zijn er diverse hulpmiddelen ontwikkeld waarmee op een gebruikersvriendelijkere manier HTML pagina's in elkaar kunnen worden gesleuteld.

Een voorbeeld van dergelijke hulpmiddelen zijn de HTML editors. Dit zijn interactieve applicaties waarmee onder besturing van een syntax editor HTML documenten kunnen worden gemaakt. De HTML editor kent de HTML syntax en voorkomt syntaxfouten in het document. Meestal werken die HTML editors volgens de WYSIWYHYG ("what you see is what you hope you get") methode. Met enkele oogopslagen is duidelijk hoe het document ongeveer door een gemiddelde browser zal worden weergegeven.

Ook zijn er een aantal zogenaamde HTML convertors ontwikkeld. Dit zijn hulpmiddelen die de bestanden van bestaande tekstverwerkings- en andere applicaties kunnen converteren naar HTML. Hierdoor kunnen reeds gemaakte documenten naar HTML worden geconverteerd ter voorbereiding van de publicatie van die informatie op het World Wide Web. De bekendste convertors zijn de Internet Assistents van Microsoft waarmee verschillende Office bestanden in HTML kunnen worden weggeschreven.

Het meest complete hulpmiddel wordt gevormd door de "web design tools". Het betreft hier applicaties waarmee volledige netwerken van hypermedia documenten kun-

nen worden ontwikkeld. Een bekend voorbeeld hiervan is FrontPage van Microsoft (ingekocht van Vermeer Technologies). Met FrontPage kan bijvoorbeeld een heel web van documenten worden ontwikkeld. FrontPage bewaakt onder andere de consistentie in opmaak van, en verwijzingen tussen, de documenten.

4.10 HTTP

HTTP

HyperText Transport Protocol
Functies:
- Ophalen documenten
- Opsturen ingevulde formulieren

Text based command/response protocol
Commandoís: GET, HEAD, POST
Well known port: 80
Versies: HTTP/0.9 en HTTP/1.0
Stateless

Notities

De meeste transacties tussen WWW clients en servers worden uitgevoerd volgens het HTTP protocol. HTTP is een lichtgewicht protocol voor het ophalen van documenten en opsturen van informatie. Het is specifiek ontwikkeld voor toepassing in het World Wide Web. De client zijde van het HTTP protocol wordt ingevuld door de HTTP browsers. Op de HTTP server draait een HTTP server applicatie welke de verbindingen opneemt en afhandelt.

Het HTTP protocol is volgens de beste Internet tradities een tekst geörienteerd command/response protocol (net als SMTP en POP). Dit heeft tot gevolg dat HTTP sessies met telnet kunnen worden nagespeeld om de werking van een server te testen. De commando's van het HTTP protocol zijn GET (om een document op te halen), HEAD (om alleen de headers van document te krijgen) en POST (om informatie naar de server op te sturen). Het HTTP protocol maakt gebruik van de well known port 80. De eerste HTTP implementaties hadden nog slechts een beperkte functionaliteit. Deze implementaties staan heden ten dage bekend als HTTP 0.9. Alle hedendaagse HTTP clients en servers ondersteunen de volledige variant HTTP 1.0. Aan de standaardisering van HTTP 1.1 wordt momenteel gewerkt.

Kenmerkend voor HTTP is dat het een toestandsloos protocol is. Dit houdt in dat de HTTP server geen informatie onthoudt van verbinding tot verbinding. Iedere request wordt op zichzelf beoordeeld, zonder referentie aan eerdere requests. HTTP kent dan ook het begrip "sessie" absoluut niet. Iedere keer dat een document wordt opgehaald of informatie wordt opgestuurd bepaalt de HTTP server of die transactie is toegestaan.

HTTP vereist verder ook geen authenticatie. De browsers hoeven normaal gesproken niet aan te loggen om een document op te kunnen halen. In het HTTP protocol zijn

slechts marginale functies aanwezig voor het bepalen van de identiteit van de gebruiker.

4.11 HTTP server software

HTTP server software

NCSA httpd
CERN httpd
Microsoft Internet Information Server
WebSite
Netscape Server
Apache
Stronghold
OSP httpd

Notities

Op de slide staan voorbeelden van diverse HTTP server implementaties. Er bestaan een groot aantal public domain implementaties van HTTP servers. Hiervan zijn de NCSA, CERN en Apache servers het bekendst. Daarnaast hebben een aantal bedrijven ook commerciële HTTP servers ontwikkeld.

4.12 HTTP voorbeeld

HTTP voorbeeld

```
GET / HTTP/1.0
From: josv@osp.nl
... overige headers
                                        <== Lege regel = scheider
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 391
... overige headers

<html><head><h1>DutchWorks Seminar</h1></head>
<body><h1>Voorbeeld pagina</h1>
... meer html
</html>
```

Notities

Zoals gezegd is HTTP een “traditioneel” text based command/response protocol. Op de slide staat een met telnet uitgevoerd voorbeeld van HTTP transactie (telnet naar poort 80 van een WWW server). De client opent de verbinding en geeft een GET commando. In dit GET commando wordt (onder andere) ook aangegeven wat de URI is van het document dat moet worden teruggegeven. Naast het GET commando stuurt de client ook nul of meer request headers mee. De server weet dat de request compleet is ontvangen zodra de eerste lege regel binnenkomt.

De server spoort vervolgens het gevraagde document op en stuurt het terug. De server response begint met een “HTTP/1.0 200 OK” regel om aan te geven dat het gevraagde document beschikbaar is. Vervolgens stuurt de server nul of meer response headers. Hierin wordt onder andere aangegeven wat het documenttype van het gevraagde document is. Na de response headers volgt een lege regel en de inhoud van het gevraagde document.

Indien een document ingesloten objecten bevat (zoals plaatjes) moet de browser meerdere HTTP GET transacties uitvoeren alvorens het gehele hypermedia document “binnen” is.

4.13 Waar komen documenten vandaan?

Waar komen documenten vandaan?

- Disk van de HTTP server
- Gegenereerd door HTTP server
  - Bijvoorbeeld directory index
- Gegenereerd door *server side* applicaties

Notities

Bij alle in gebruik zijnde HTTP servers wordt de “document store” gevormd door een gedeelte van de hard disk van de server. URI’s corresponderen in dat geval met de relatieve padnaam van een bestand in de document directory van de HTTP server. Het HTTP protocol schrijft dit echter niet voor. Er is tenminste één server waarbij de documentenverzameling wordt gevormd door een database (de HyperWave server van de universiteit van Graz). HTTP servers ondersteunen meestal ook dat URI’s verwijzen naar uitvoerbare programma’s. In het geval een browser naar een dergelijke uitvoerbare URI verwijst voert de HTTP server de URI uit en stuurt de output van dit programma door naar de browser.

4.14 HTTP, pro en contra

HTTP, pro en contra

Pro:
- eenvoudig
- stateless
- ondersteuning voor meerdere documenttypen

Contra:
- stateless
- performance: gebruikt een TCP/IP verbinding per documentdeel
- slechte beveiliging

Notities

Er is al veel gediscussieerd over het HTTP protocol. Feit is dat het protocol eigenlijk totaal niet bedoeld was voor de manier waarop het vandaag aan de dag wordt gebruikt. Veel organisaties maken gebruik van het feit dat er veel goede browsers in omloop zijn om HTTP en HTML in te zetten voor het bouwen van complexe client/server applicaties. In principe biedt HTTP hier geen features voor. Met extra software en een aantal trucjes kan het protocol wel worden opgelapt om wat meer functionaliteit te bieden, maar het blijft behelpen.

Één van de belangrijkste eigenschappen van HTTP is dat het een toestandsloos protocol is. De HTTP server beziet iedere request als een op zichzelf staand gebeuren en doet geen pogingen om informatie te onthouden van verbinding tot verbinding. Voor het implementeren van “eenvoudige” algemeen toegankelijke electronische informatiediensten is dit een voordeel. Voor meer complexe client/server transacties is de toestandsloosheid echter een groot nadeel. Met trucjes als *cookies* kunnen we deze toestandsloosheid opheffen. De verantwoordelijkheid voor het onthouden van die toestand ligt dan echter bij de applicatie. Moderne HTTP servers hebben ingebouwde mechanismen voor het onthouden van informatie van verbinding tot verbinding (zoals de OSP httpd).

HTTP kent ook weinig ingebouwde beveiligingsmethoden. Er is een vorm van authentication, de zogenaamde *Basic Authentication* waarbij gebruikers zich moeten melden met userid en password. Dit betekent echter dat vanaf dat moment bij iedere request het userid password onversleuteld met de request worden meegestuurd!.

Een ander nadeel van HTTP is de slechte performance voor complexe hypermedia documenten. Indien een complex document meerdere elementen (zoals plaatjes, video,

geluid) bevat wordt ieder element door middel van een aparte HTTP request opgehaald. Dit heeft als gevolg dat er per HTML element een aparte TCP verbinding moet worden opgezet.

Een gedeelte van de hier genoemde nadelen zal naar verwachting worden opgeheven in de volgende release van HTTP: HTTP/1.1.

Module 5

Web based applicaties

5.1 Web based applicaties

Web based applicaties

Gebruik van WWW voor transactie georienteerde applicaties.

Dynamisch genereren van documenten (HTML) door programmatuur op de HTTP server.

Notities

Veel organisaties hebben het web ontdekt als universeel interactief transactiemedium voor het realiseren van client/server applicaties. Hierbij speelt een grote rol dat de client en server applicaties van het web wijd verbreid zijn: gebruikers hoeven niet eerst een proprietary client applicatie te installeren, ze kunnen meestal gelijk aan de gang met hun huidige web browser.

Toepassingen als electronische winkels hebben naast een puur informatief gedeelte vaak ook een interactieve sectie waar goederen en dergelijke kunnen worden besteld. Dit vereist dat er interactie is tussen de gebruiker en de WWW server van de winkel. Ook remote database applicaties vereisen dit soort interactie. Steeds meer traditionele applicaties komen ook met een WWW interface zodat een gebruiker met een web browser de applicatie over het intranet of Internet kan gebruiken.

Web based applicaties zijn gebaseerd op twee fundamenten:

1. HTML invoerformulieren die door gebruikers kunnen worden ingevuld.
2. Programmatuur op de WWW server die de ingevulde formulieren verwerkt en op basis daarvan *dynamisch* weer nieuwe HTML pagina's (of formulieren) genereert.

5.2 Voorbeeld: OSP Triviant 1

Notities

Op de slide ziet u een voorbeeld van een web based applicatie. Het betreft hier een tamelijk eenvoudige applicatie voor het spelen van het spel triviant. De gebruiker moet allereerst aanloggen aan de triviant applicatie met behulp van een userid en password. De applicatie biedt vervolgens een aantal mogelijkheden: het beantwoorden van de vraag van de dag, het bekijken van de scorelijsten en het invoeren van een nieuwe vraag.

5.3 Voorbeeld: OSP Triviant 2

Notities

Het op de slide getoonde HTML document is dynamisch gegenereerd op basis van de aanloginformatie van de gebruiker.

5.4 Implementatie

Implementatie

CGI Programmatuur
NSAPI/ISAPI applicaties
Tools:
- WebObjects
- Internet Database Connector

Nieuw: Java *servlets*
Active Server Pages

Notities

Om web based applicatie te kunnen ontwikkelen dient er programmatuur aan de server zijde te worden geïmplementeerd. Deze programmatuur is verantwoordelijk voor het afhandelen van een HTTP request en het genereren van HTML pagina's op basis van deze request.

Op de slide staan een aantal mogelijkheden voor het implementeren van web based applicaties. De simpelste (en gestandaardiseerde) methode hiervoor is CGI-programmatuur. Deze mogelijkheid wordt door alle HTTP server applicaties ondersteund. Daarnaast kennen verschillende HTTP servers hun eigen mogelijkheden voor het implementeren van web applicaties.

5.5 CGI

CGI

Common Gateway Interface

URL refereert een programma i.p.v. een document

HTTP server software start dit programma

Input : Allerhande informatie (variabelen)

Output : Naar de browser

CGI programmaís meestal in C, C++ of scripttalen (Perl, shell script)

Notities

De oudste manier om web applicaties te bouwen is via het *Common Gateway Interface*. Dit is een standaard volgens welke de HTTP server programma's opstart en de gegevens uit het originele request meegeeft. Het is de taak van het CGI programma om de request te verwerken en op basis van de gegevens uit de request (bijvoorbeeld de meegegeven waarden van de invulvelden) een nieuw document te genereren. De uitvoer van het CGI programma wordt als antwoord naar de client gestuurd. Het triviant voorbeeld van de voorvorige slide is met behulp van CGI programma's geïmplementeerd.

De werking van CGI programma's is tamelijk transparant voor de WWW client. Een CGI programma wordt gestart door middel van een URI die door de HTTP server wordt herkend als een uitvoer programma. Hoe de HTTP server dit beslist is niet universeel vastgelegd. Iedere HTTP server leverancier mag daar zijn eigen standaard voor kiezen. Meestal is gevoelsmatig aan de URL wel te zeggen of het een CGI programma betreft of een gewoon document. Veel sites maken er een gewoonte van om CGI programma's in een aparte /cgi-bin directory op de HTTP server te plaatsen.

CGI programma's kunnen in iedere gewenste programmeertaal worden ontwikkeld. Een erg populaire taal is de UNIX script taal 'perl', maar ook C, C++ en python zijn populaire talen voor het bouwen van CGI programma's. Een belangrijk aandachtspunt bij CGI programma's is beveiliging. Door middel van het CGI mechanisme zijn browsers immers in staat om programma's (eventueel met parameters) op te starten op de HTTP server. Deze programma's draaien daar met de rechten van de HTTP server applicatie. Indien er in de CGI programmatuur bugs zitten kunnen die wellicht worden uitgebeuit door hackers. Het niet meegeven van verwachte argumenten, of het meegeven van onverwachte waarden in de argumenten is een bekende aanvalsmethode.

Een moeilijk probleem bij CGI programma's is dat de HTTP server geen toestand onthoudt van verbinding tot verbinding. Dit betekent dat een applicatie die bestaat uit meerdere CGI programma's zelf een methode moet implementeren om informatie te onthouden. Ook is niet bekend of een gebruiker zijn browser heeft gestopt, dus de onthouden informatie moet ook regelmatig worden opgeschoond.

5.6 NSAPI / ISAPI

NSAPI / ISAPI

Netscape Server API

Internet Server API (Microsoft)

Serverzijde applicaties in *shared libraries* of DLLís die door de HTTP server worden geladen

Onderscheppen HTTP request

Genereren HTTP respons (HTML)

Geschreven in C / C++

Notities

Een nadeel van CGI programmatuur is dat het losstaande programma's betreft die iedere keer door de HTTP server worden gestart zodra er een request binnenkomt. Indien deze programma's informatie moeten bijhouden doen ze dit meestal door middel van bestanden op de disk. Het is niet mogelijk dit in het geheugen te doen omdat het CGI programma iedere keer stopt zodra er een transactie is afgerond. Het veelvuldig starten en stoppen van CGI programma's heeft ook geen erg gunstig effect op de performance van het systeem.

Om deze reden hebben Netscape en Microsoft een manier bedacht om web applicaties als componenten te laten draaien in de HTTP server applicatie. De Netscape Server ondersteunt hiervoor de NSAPI; Microsoft's Internet Information Server kent de ISAPI. Beiden zijn programmeursinterfaces waarmee applicaties in C/C++ kunnen worden geschreven en *in* de server applicatie kunnen worden geladen (via zogenaamde *shared libraries* of *shared objects*. Bij het starten van de HTTP server laadt deze automatisch alle geconfigureerde applicatiemodules in het geheugen. Op het moment dat er een HTTP request binnenkomt krijgen de geladen applicaties de mogelijkheid om deze request af te handelen. Als zij dit doen zijn ze ervoor verantwoordelijk om het antwoord naar de client te genereren (HTML).

In tegenstelling tot CGI programma's zijn NS/IS API applicaties modules die in het interne geheugen van de server informatie kunnen onthouden met betrekking tot de transacties. Verder hoeven de applicaties niet iedere keer te worden geladen. Hierdoor is de performance van NS/IS API applicaties beter dan die van CGI programma's. Een nadeel van dit soort applicaties is dat ze gebonden zijn aan één HTTP server applicatie. Een ISAPI applicatie kan alleen maar draaien in de HTTP server van Microsoft. CGI is daarentegen een standaard die door alle HTTP servers wordt ondersteund.

5.7 Hulpmiddelen

Hulpmiddelen

- NeXt WebObjects
  - Ondersteuning voor serverzijde applicaties
  - Toegang tot databases
  - Bijhouden van *state*
  - Faciliteiten voor genereren van HTML
- Microsoft Internet Database Connector
  - ISAPI DLL
  - Genereren van HTML uit ODBC databases
  - Invoer in database uit HTML formulieren

Notities

Ter ondersteuning van het ontwikkelen van web applicaties zijn er door diverse leveranciers hulpmiddelen op de markt gebracht ter ondersteuning van het implementeren van dergelijke applicaties. Een aantal voorbeelden hiervan zijn WebObjects van NeXt en de Merchant Server van Microsoft. Deze hulpmiddelen vergemakkelijken meestal het genereren van HTML pagina's, het bijhouden van toestandsinformatie en de toegang tot diverse databasesystemen.

Daarnaast krijgen steeds meer server applicaties componenten voor het functioneren in een WWW omgeving. Zo kunnen moderne databasesystemen bijvoorbeeld automatisch HTML genereren zodra er iets in de database wijzigt (bijvoorbeeld produktinformatie). De bekende workgroup applicatie Lotus Notes heeft zelfs een hele HTTP server component gekregen voor het toegankelijk maken van de gehele document database via het World Wide Web.

5.8 Java servlets

Java servlets

Mogelijkheid om Java programmatuur op de server te draaien.

HTTP server bevat Java *Virtual Machine*

Conceptueel gelijk aan NSAPI / ISAPI

Notities

Een nieuwe ontwikkeling op het gebied van web applicaties is de mogelijkheid om web applicatie te implementeren met Java programmatuur op de server zijde. Technisch gesproken is dit vrijwel hetzelfde als het implementeren van een WWW applicatie met een in C/C++ geschreven NS/IS API applicatie. In het geval van Java *servlets* is de applicatie geschreven in Java. De HTTP server bevat een Java *virtual machine* voor het uitvoeren van deze Java programmatuur. De in Java geschreven applicatie krijgt van de HTTP server de kans om HTTP requests te onderscheppen en te interpreteren.

5.9 Active Server Pages

Active Server Pages

Microsoft Internet Information Server
¡Serverzijde Visual Basicî
Andere scripttalen ook mogelijk (Perl)
ActiveX objecten voor sessies/connecties

Notities

De Microsoft Internet Information server heeft de mogelijkheid om serverzijde applicaties te ontwikkelen via zogenaamde “Active Server Pages”. Deze ASP’s zijn programmaatjes geschreven in een of andere script taal (bijvoorbeeld Visual Basic) welke door de web server worden gestart. Deze programma’s verwerken de parameters uit de request en genereren HTML als antwoord. Praktisch gesproken is dit vrijwel hetzelfde als NSAPI/ISAPI/Java servlets.

5.10 Web applicaties, pro en contra

Pro en contra

Pro
- Eenvoudig voor simpele toepassingen
- Weinig/geen eisen aan de client (browser)
- *Cool!*

Contra
- Complex voor moeilijkere toepassingen
- In principe stateless
- Beveiliging
- Internet is geen gegarandeerd communicatiemedium

Notities

In principe is het World Wide Web bedacht voor het electronisch ter beschikking stellen van informatie en documentatie. Ter ondersteuning hiervan zijn simpele transactiemogelijkheden geïmplementeerd. Veel organisaties “misbruiken” het web echter als vehikel voor complexere client/server applicaties. De in het WWW toegepaste protocollen en technieken bieden hier eigenlijk geen ondersteuning voor. Dit betekent dat de programmeur van een complexe web applicatie een behoorlijk grote toverdoos met trucjes moet hebben om de ontbrekende (maar benodigde) features zelf te programmeren. Inmiddels zijn diverse toverdozen te koop voor het implementeren van web applicaties.

Een groot voordeel van het ter beschikking stellen van een applicatie via het web (Internet of intranet) is dat de benodigde client software in ruime mate voorhanden is op alle denkbare (en niet denkbare) platforms. Hierdoor is de applicatie zonder al te veel problemen breed toegankelijk. Ook in grote organisatie met een heterogeen intern netwerk kan het veel voordelen opleveren om een applicatie via WWW technologie ter beschikking te stellen. Alhoewel nog niet te zeggen valt hoe de toekomstige ontwikkelingen op het gebied van Web-TV en Network Computers zullen verlopen is het investeren in een Web applicatie (gezien de reeds aanwezige kritieke massa) een betrekkelijk veilige aangelegenheid.

Module 6

Active Content

6.1 “Plain Vanilla” WWW

Plain Vanilla WWWXLLM

Domme client
  Vindt geen applicatieverwerking plaats
Slimme server
  In flight genereren van documenten
    HTML
    GIF

Notities

In de originele WWW gedachte bestond het web uit relatief slimme servers die documenten konden ophoesten (of genereren) en domme clients die slechts de taak hadden de opgeleverde documenten te tonen. In die basisgedachte was het niet mogelijk voor een WWW server om een stukje programmatuur op de client uit te (laten) voeren. De oudere web browsers, zoals lynx en Mosaic, vormen een perfecte implementatie van die gedachte. Het grote voordeel hiervan is dat er aan de kant van de WWW client niet veel rekenkracht of opslagcapaciteit aanwezig hoeft te zijn.

6.2 Nadelen van "plain vanilla" WWW

Nadelen van *plain vanilla* WWWLLI

Geen verwerking aan de client zijde, dus:
- Controle op fouten op de server (loze transactie)
- Beperkte animatie in de documenten
- Mogelijkheden beperkt door de browser
  - Documenttypen
  - Acties (zoals chipcard betalingen)

Notities

Webmasters zochten echter al snel naar wegen om de statische HTML pagina's te voorzien van bewegende elementen. Ook het toenemende gebruik van het web voor transactie georienteerde web applicaties riep om meer kracht en flexibiliteit aan de kant van de browser. Sites wilden graag de mogelijkheid om:

- Animaties uit te voeren op de client.
- Controle op invulvelden uit te voeren op de client *voordat* deze naar de server worden gezonden.
- De browser uitbreiden met mogelijkheden voor het tonen van niet standaard ondersteunde documenttypen.
- Acties (zoals betalingen en dergelijke) uit te kunnen laten voeren op de client.

.

Om dit mogelijk te maken dient de browser te worden uitgebreid met extra functionaliteit.

6.3 Uitbreiding van de browser

Uitbreiding van de browser

Helper applicaties
Plug-Ins
JavaScript
VB Script
Java
ActiveX

Notities

Om te ontkomen aan de beperkingen van de “domme’ WWW client dient deze te kunnen worden uitgebreid met extra functionaliteit. Met die extra functionaliteit kunnen statische pagina’s “tot leven worden gewekt”, en kunnen niet-standaard acties aan de client worden geïnitieerd.

Een nadeel van dergelijke uitbreidingen aan de client zijde is dat daarmee afhankelijkheden worden geïntroduceerd naar de configuratie van de client. Een web site die gebruik maakt van dergelijke uitbreidingen kan er meestal niet meer vanuit gaan dat iedere web browser de mogelijkheden heeft om de web applicatie(s) in kwestie te ‘draaien’. Meestal bieden dit soort web sites dan ook de mogelijkheid om de benodigde extra programmatuur te downloaden.

Op de slide zijn de momenteel in zwang zijnde mogelijkheden voor ‘Active Content’ opgesomd. In de nuvolgende slides worden deze één voor één besproken.

6.4 Helper applicaties

Notities

De meest eenvoudige methode om de functionaliteit van de browser uit te breiden is de ‘Helper’ applicatie. Het betreft hier een programma dat door de browser wordt gebruikt om documenttypen weer te geven die niet standaard door de browser worden begrepen. Zoals eerder in de shop is verteld kan een browser op een of andere manier bepalen wat het MIME bestandstype is van het opgehaalde document. Dit gebeurt aan de hand van de bestandsnaam van het opgehaalde document of aan de hand van de ‘Content-Type’ header in de HTTP response. Indien de browser het bestandstype ‘kent’ kan hij het zelf op een bepaalde manier tonen.

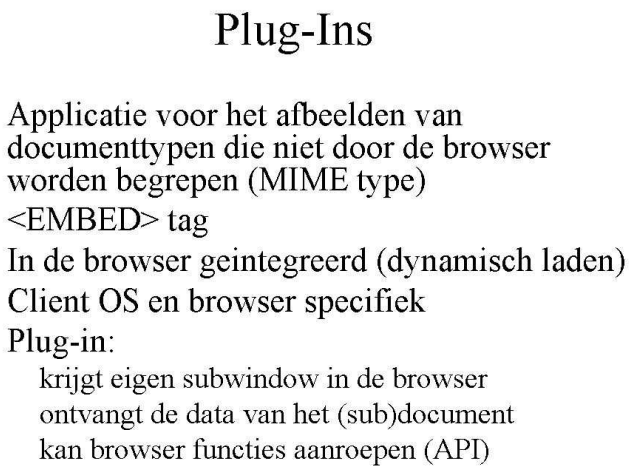
Voor documenttypen die de browser niet kent kan deze worden geconfigureerd om een helper applicatie op te starten. In de configuratie van de browser wordt dan aangegeven dat voor documenttype ‘abc/xyz’ de helper applicatie ‘zus-en-zo’ moet worden gestart. In het geval de browser een document van het gespecificeerde type ophaalt slaat hij dit document op en start hij de helper applicatie. Het is dan de verantwoordelijkheid van de helper applicatie om het document in kwestie af te beelden.

Aangezien de helper applicatie een losstaand programma is vindt er na het starten van de helper door de browser geen communicatie meer plaats tussen helper en browser. De helper heeft in principe de beschikking over de volledige mogelijkheden van de client, en kan dus netwerkverbindingen openen, met de gebruiker communiceren en bestanden manipuleren.

Helper applicaties hoeven niet specifiek voor het World Wide Web te worden ontwikkeld. Standaard applicaties kunnen ook als helper worden ingeschakeld. Zo kan de Word Viewer applicatie van Microsoft ook worden gebruikt als helper applicatie voor het weergeven van Word documenten die via het World Wide Web worden opgehaald.

Helper applicaties hoeven ook niet speciaal voor een bepaalde browser te worden ontwikkeld. De enige interactie tussen de browser en de helper is dat de browser op de voor het client-OS geldende standaardmanier wordt gestart.

6.5 Plug-Ins



Notities

Een andere mogelijkheid om de functionaliteit van een browser uit te breiden is door middel van plug-ins. Een plug-in is een soort helper applicatie die *in* de browser wordt opgestart. Als output windows krijgt een plug-in een gedeelte van het window van de browser toegewezen. Dit betekent dat voor de gebruiker de plug-in een integraal onderdeel uitmaakt van het getoonde HTML document. Plug-ins hebben echter een eigen 'leven', ze kunnen onafhankelijk van de browser allerlei aktiviteiten ontplooien. Plug-ins kunnen ook met de browser communiceren via een programmeur interface (API). In tegenstelling tot een helper applicatie is een plug-in geschreven voor een specifieke browser en client-OS.

Plug-ins worden geaktiveerd door de EMBED tag. De browser die de HTML interpreteerd maakt een subwindow voor de plug-in, laadt de plug-in, download de plug-in data en start de plug-in.

De integratie tussen een plug-in en de browser is optimaal. Dientengevolge ziet het getoonde document er zeer gelikt uit. Het niet standaard onderdeel wordt integraal in het document getoond in het door de plug-in aangestuurde subwindow. Een belangrijk nadeel van een plug-in is dat de plug-in apart op de computer van de gebruiker moet zijn geïnstalleerd. Verder zullen documenten waarin gebruik wordt gemaakt van de EMBED tag (om een plug-in te aktiveren) alleen werken gebruikers die de plug-in ook hebben geïnstalleerd. Veelal bevatten sites die gebruik maken van plug-ins ook mogelijkheden om de gebruikte plug-ins te downloaden.

6.6 JavaScript

JavaScript

Voorheen *LiveScript*
Ondersteund door Netscape en Microsoft IE
Script programmeertaal
Source gaat mee in het document via <SCRIPT> tag
Script wordt door de browser uitgevoerd
Beperkte functionaliteit

Notities

Een andere moderne mogelijkheid van het uitvoeren van programmatuur aan de client zijde is het meesturen van die programmatuur in het HTML document. De bedenker van dit fenomeen is de Amerikaanse leverancier van WWW technologie Netscape. De uitvinding van Netscape, LiveScript (later hernoemd naar JavaScript), maakt het mogelijk om met het HTML document allerlei programmatuur (in source code) mee te sturen. De ontvangende browser interpreteert naast de HTML ook de JavaScript source en voert die uit.

JavaScript bevat allerlei mogelijkheden voor het afvangen van gebruikersinteractie (muisbeweging, tekstinvoer, drukken op knopjes) en het automatisch uitvoeren van programmatuur zodra een pagina wordt geladen. Hierdoor kan een web applicatie allerlei JavaScript code met een document meesturen om een gedeelte van de applicatieverwerking aan de client zijde uit te laten voeren. Hierdoor kunnen bijvoorbeeld controles op ingevulde velden al door de browser worden uitgevoerd alvorens het hele formulier naar de server wordt gestuurd. Verder kunnen JavaScript procedures allerlei berekeningen uitvoeren en het resultaat daarvan terugplaatsen in het document.

JavaScript biedt geen algemene functionaliteit voor het benaderen van bestanden of het netwerk. Hierdoor brengt het uitvoeren van JavaScript programmatuur door de browser geen onaanvaardbare beveiligingsrisico's met zich mee. Een belangrijk nadeel van JavaScript is natuurlijk dat WWW pagina's die JavaScript code bevatten alleen werken als de browser waarmee die pagina's worden bekeken ook JavaScript compatibel is. Non-JavaScript browsers geven vaak foutmeldingen indien ze een JavaScript script tegenkomen of laten slechts een halve pagina zien.

6.7 VBScript

VBScript

Visual Basic Script
Ondersteund door Microsoft IE
A la JavaScript

Notities

Microsoft kon zich natuurlijk met het aanstormende Java en JavaScript geweld niet onbetuigd laten en het heeft dan ook zijn standaard macrotaal Visual Basic geschikt gemaakt voor het World Wide Web. het resultaat hiervan is VBScript, een geïnterpreteerde programmeertaal die net als JavaScript met de HTML documenten mee kan worden gestuurd (ook via de SCRIPT tag). VBScript is qua toepassing en implementatie vrijwel gelijk aan JavaScript. VBScript wordt echter bij mijn weten zo goed als nergens toegepast.

6.8 Java

Java

Hype, Hyper, Java
Object georienteerde programmeertaal
Complete programmeertaal (vgl: C++)
Ontwikkeld door Sun Microsystems
Java compiler vertaalt naar architectuur neutrale bytecode (pseudo machine taal)
Breed omarmd door IT-industrie
WWW Toepassing: *servlets* en *applets*

Notities

De grootste Internet hype van de laatste eeuw is zonder meer Java. Deze programmeertaal van leverancier Sun heeft in 'no time' een zeer aanzienlijke schare aanzienlijke fans verkregen. Grote IT leveranciers als IBM, HP, Netscape en Microsoft kondigden vrijwel gelijk met de vrijgave van Java aan de programmeertaal in meer of mindere mate te gaan ondersteunen.

Java is een door Sun ontwikkelde object geörienteerde, volledige, platform onafhankelijke, *multi threading*, robuuste, veilige, architectuur neutrale programmeertaal. De Java vertaler zet Java source code om in een neutrale tussencode (P-code, bytecode) die door een run time interpreter (de zogenaamde Java Virtual Machine) moet worden geïnterpreteerd. Java is een moderne en goede programmeertaal waarin zonder problemen grotere en kleinere applicaties kunnen worden ontwikkeld. Java bevat ingebouwde mogelijkheden voor bestandsmanipulatie, het opbouwen van netwerkverbindingen en het bouwen van grafische gebruikers interfaces.

De grote populariteit van Java in Internet is ontstaan doordat Sun in Java een browser ontwikkelde (HotJava) die het mogelijk maakte om stukken gecompileerde Java programmatuur (Java executables) over het netwerk te laden en in de browser uit te voeren. We noemen deze gedownloade Java programmatuur ook wel 'applets'.

6.9 Java applets

Java applets

Soort dynamische plug-in

<APPLET> tag

- Browser download Java ë.classí file (bytecode)
- Applet krijgt eigen subwindow
- Browser voert Java executable uit

Applet kan:

- Tekenen (animatie)
- User interactie
- Netwerkverbindingen opbouwen (remote databases)

Notities

Een Java applet is een soort dynamische gedownloade plug-in. De HTML programmeur geeft in zijn document aan dat op een bepaalde plek in het document een Java programma moet draaien. Hij/zij geeft dit aan door het specificeren van een APPLET tag. De browser die dit document interpreteert herkent de APPLET tag, downloadt de Java executable (in de platform onafhankelijk bytecode) met het HTTP protocol en interpreteert deze bytecode. De draaiende Java applet krijgt net als een plug-in een eigen subwindow in de browser. Hierdoor voert de applet *in* de browser uit. Voor de gebruiker is de draaiende applet dus een integraal onderdeel van het document. Conceptueel kan een Java applet dus worden gezien als een kruising tussen een plug-in en een JavaScript script.

Een applet heeft de kracht van een plug-in (want geschreven in een volledige, krachtige programmeertaal) en de dynamiek van een JavaScript script (want gedownload over het netwerk). De gebruiker hoeft dus niet eerst een apart stuk programmatuur te installeren voordat een Java applet kan gaan draaien! De applet wordt ‘on demand’ over het netwerk opgehaald en uitgevoerd. De mogelijkheden hiervoor zijn natuurlijk eindeloos! Browsers kunnen dynamisch worden uitgebreid met de mogelijkheden om bepaalde typen documenten te tonen of om bepaalde acties te ondernemen (zoals het uitvoeren van een betaling).

Het over het netwerk ophalen en zonder meer uitvoeren van een programma is natuurlijk beveiligingstechnisch een hachelijke zaak. Om deze reden hebben de bedenkers van Java bewust allerlei faciliteiten niet in de taal geïmplementeerd (zoals pointers en het rechtstreeks kunnen benaderen van geheugen). De Java bytecode interpreters in de browsers laten vervolgens een groot aantal zaken niet toe. Zo kunnen Java applets

geen bestanden benaderen op de lokale disks van de gebruiker en kunnen ze alleen maar netwerkverbindingen opbouwen met de site waarvandaan ze gedownload zijn. De Java applet draait als het ware in een software gevangenis waaruit ze niet kunnen ontsnappen.

Alhoewel er intussen meerdere Java gerelateerde beveiligingsbugs boven water zijn getoverd zijn deze allemaal terug te voeren op fouten in de implementatie. In het Java beveiligingsmodel zijn nog geen structurele fouten aangetoond!

6.10 Java, pro en contra

Pro en contra

Pro:

- Dynamische inhoud, programma wordt op de client uitgevoerd
- Krachtige programmeertaal
- Automatische software distributie
- *Mega cool!*

Contra:

- Java *enabled* browser nodig
- Performance (download en uitvoering)
- Beveiliging

Notities

Java wordt door velen gezien als de toekomst van Internet en intranet. De mogelijkheid om applicaties in een architectuur neutrale tussencode op te sturen naar een client om daar te worden uitgevoerd is natuurlijk ook ongekend krachtig. Het maakt niet uit wat voor soort computer de gebruiker heeft, als zijn/haar browser Java ondersteunt kan de Java applet ter plekke worden uitgevoerd. Java is intussen zelfs doorgedrongen op de server waar web applicaties kunnen worden ontwikkeld in Java. Deze Java servlets draaien natuurlijk niet in een software gevangenis en kunnen alle mogelijkheden van het systeem benaderen. Er wordt door diverse organisaties grootscheeps geïnvesteerd in Java en het is ontegenzeggelijk dat deze technologie een grootste toekomst tegemoet gaat.

In de hedendaagse omgeving brengt het gebruik van Java nogal wat nadelen met zich mee. Allereerst vereist Java natuurlijk een browser die Java bytecode begrijpt en kan uitvoeren. Op dit moment zijn er nog geen Java enabled browsers beschikbaar voor DOS en Windows 3.x. Verder is de uitvoering van Java applets bij tijd en wijle nog ontzettend traag. Dit wordt veroorzaakt door de tijd benodigd voor het downloaden van de applet en de vertraging die het interpreteren van de tussencode met zich meebrengt. Nieuwere en betere technologie (zoals snellere netwerkverbindingen en het *Just-In-Time* omzetten van de Java bytecode naar echte machinetaal) zullen deze nadelen naar verwachting snel opheffen.

Daarnaast is en blijft beveiliging een belangrijk aandachtspunt.

6.11 ActiveX

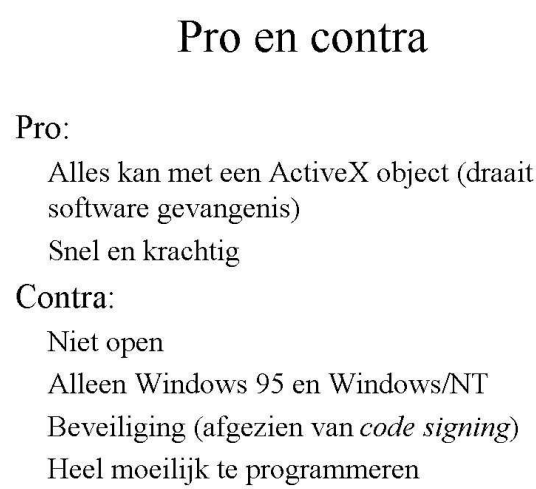
ActiveX

Microsoft standaard
Afgeleid van OLE
Alleen voor 32-bit Windows OSíen
  Windows 95, Windows/NT
Downloaden en uitvoeren Windows
executable als onderdeel van de browser
<OBJECT> tag

Notities

Microsoft's antwoord op Java is ActiveX. Net als Java betreft het hier een faciliteit voor het over het netwerk ophalen en in de browser uitvoeren van programma's. In tegenstelling tot Java betreft het echter bij ActiveX geen executables in een neutrale tussencode maar in echte machinetaal. Omdat deze executables rechtstreeks door de processor van de klant kunnen worden uitgevoerd is de performance van ActiveX objecten aanzienlijk beter dan die van Java applets.

6.12 ActiveX, pro en contra



Notities

ActiveX objecten draaien in tegenstelling tot Java applets niet in een software gevangenis en hebben dus volledige toegang tot alle faciliteiten van de client computer. Uit beveiligingsoogpunt kunnen ActiveX controls worden gesigneerd door de leverancier zodat je zeker weet van wie de module afkomstig is. Desalniettemin is de beveiliging van ActiveX controls een “hot item”.

Alhoewel Microsoft zijn uiterste best doet om ActiveX tot algehele Internet standaard te verheffen is ActiveX momenteel volledig verbonden met de 32-bits Microsoft besturingssystemen Windows 95 en Windows/NT. Hierdoor kunnen pagina’s met ingesloten ActiveX controls alleen worden bekeken op client computers met een Win32 besturingssysteem.

Module 7

Overige onderwerpen

7.1 Do you want a cookie?

Do you want a cookie?

Netscape cookies

Breed ondersteund

Manier om state vast te houden in een web applicatie

Server (CGI-programma) genereert cookie

Client presenteert cookie bij iedere request

Notities

Zoals reeds vele malen in deze shop aan de orde is gekomen is het HTTP protocol toestandsloos: de HTTP server onthoudt geen informatie van verbinding tot verbinding. Voor web applicaties is dit een groot nadeel. Netscape heeft ter verlichting van de problemen een mechanisme geïntroduceerd waarmee web applicaties in samenwerking met WWW browsers toestand kunnen onthouden van verbinding tot verbinding. Deze zogenaamde *Netscape cookies* worden inmiddels door een groot aantal web browsers ondersteund.

Applicaties die toestandsinformatie willen onthouden hebben bij volgende verbindingen van dezelfde client een identificerend element nodig om de opgeslagen (of onthouden) toestand weer te aktiveren. Het is namelijk in het web zeer wel mogelijk dat een applicatie door zeer veel gebruikers tegelijk wordt gebruikt. In dat geval is het nodig dat die applicatie van meerdere sessies tegelijk de toestand kan bewaren.

Op het moment dat de applicatie de eerste transactie van een nieuwe sessie ontvangt (meestal via het login scherm van de web applicatie) genereert de applicatie een uniek *cookie*. Zo'n cookie is een tekststring van onbepaalde lengte. Meestal bevat een cookie de datum, tijd en gebruikersnaam die met de sessie worden geassocieerd. De web applicatie geeft de cookie vervolgens terug aan de browser in de vorm van een HTTP response header "Set-Cookie". De browser zal vervolgens bij iedere request naar die server het cookie weer meesturen in een HTTP request header. De web applicatie kan het aldus ontvangen cookie gebruiken om de toestand van de sessie op te zoeken en de request te verwerken.

Cookies kennen ook een vervaldatum. Als deze is verstreken wordt de cookie niet meer meegezonden en kan (moet) de web applicatie (als hij daar zin in heeft) een nieuw coo-

kie genereren. Cookies kunnen ook worden toegepast om een gebruiker onmiddelijk weer te herkennen om bijvoorbeeld speciale aanbiedingen te doen of persoonlijke instellingen te herstellen. Cookies worden meestal door de browser opgeslagen op de disk van de gebruiker.

7.2 VRML

VRML ¸Î9◊

Virtual Reality Modelling Language

Markup language voor 3D werelden

Browser staat gebruiker toe om in 3D wereld rond te wandelen

Meestal geimplementeerd als plug-in of helper applicatie

Moeilijk te ontwikkelen

Notities

Een “hot item” van enige tijd geleden, maar intussen weer wat minder prominent aanwezig, is de *Virtual Reality Modelling Languae*, VRML (of, zoals collega Theo de Ridder zegt: Frummel). VRML is een opmaaktaal voor driedimensionale werelden. In een ASCII bestand (een zogenaamd WRL bestand) geven we met VRML instructies weer hoe de wereld in elkaar zit, waar bepaalde objecten zich bevinden en waar het licht vandaan komt. VRML interpreters staan toe dat de gebruiker zich met het toetsenbord of met de muis door de 3D wereld beweegt.

In het World Wide Web worden VRML werelden geïnterpreteerd door helper applicaties of plug-ins. Er zijn zelfs enkele implementaties die VRML, Java en HTML met elkaar verweven tot een driedimensionaal, interactief informatielandschap. Het belangrijkste probleem met VRML is dat het weergeven van een grote, realistische, wereld tamelijk moeilijk is voor zowel de VRML-programmeur als de client computer. VRML zou op zich ideaal zijn voor het weergeven van een electronische winkel of het opzoeken van informatie in een virtuele bibliotheek.

7.3 HyperWave

HyperWave

Next Generation WWW
Universiteit van Graz
Uitbreidingen op WWW:
- versiebeheer
- link management
- toegangscontrole
- zoekmogelijkheden

Aparte client en server nodig

Notities

Terwijl de meeste inwoners van de wereld nog niet toe zijn aan het gebruik van het hedendaagse World Wide Web wordt aan de universiteit van Graz al vrolijk geknutseld aan het web van de toekomst. Aldaar is onder de bezielende leiding van Hermann Mauer de opvolger van het World Wide Web ontwikkeld: HyperWave (voorheen: HyperG). HyperWave stelt zich tot doel een groot aantal van de problemen van het World Wide Web op te lossen. Zo kennen HyperWave browsers en servers faciliteiten voor versiebeheer van documenten, het beheren van hyperlinks (geen blinde links meer) en ingebouwde zoekmogelijkheden.

Om HyperWave te kunnen bedrijven is een speciale client (browser) en server benodigd. Deze server gedraagt zich ook als een normale HTTP server en kan dus ook door standaard browsers worden benaderd, maar deze kunnen natuurlijk niet de verbeterde HyperWave faciliteiten gebruiken. De universiteit van Graz heeft een HyperWave server en een aantal HyperWave clients ontwikkeld.

7.4 Web robots

Web robots

Programmaís die het web onderzoeken
- Web slurping (htget)
- Zoeken
- Indexeren

Notities

Een ander fenomeen in het World Wide Web is het bestaan van zogenaamde *webbots*, ook wel “web robots” genoemd. Het betreft hier programma’s die contact opnemen met HTTP servers en geautomatiseerd documenten ophalen en verwerken. Dergelijke programma’s zijn bijvoorbeeld in staat om hele web sites te kopiëren, documenten te vinden die voldoen aan een bepaald sleutelwoord of om indexen op te bouwen van documenten. De bekende zoek servers van Yahoo en Lycos gebruiken onder andere web robots om hun informatie te verzamelen. Iedere site kan in zijn root directory een bestand met de naam “robots.txt” opnemen. In dit bestand kunnen commando’s worden opgenomen die alle of bepaalde robots de toegang tot de site geheel of gedeeltelijk afraden. De meeste web robots houden zich hieraan.

Nieuwe ontwikkelingen zijn intelligente web robots die er door hun meester op uit worden gestuurd om alle documenten te vinden die aan bepaalde voorwaarden (sleutelwoorden) voldoen. Deze programma’s draaien volcontinu geven een waarschuwing zodra ergens nieuwe documenten zijn gevonden die aan de criteria voldoen. Op die manier kunnen gebruikers bijvoorbeeld automatisch worden geattendeerd op nieuwe ontwikkelingen op een bepaald gebied.

7.5 Proxy servers

Proxy servers

Intermediairs in de WWW transacties
Maken lokale kopieen van documenten
Brengt het aantal Internet verbindingen terug
Kans op (ietwat) verouderde informatie

Notities

De praktijk wijst uit dat het vaak voorkomt dat gebruikers in een organisatie of van een provider regelmatig dezelfde documenten ophalen. De individuele browsers hebben hier natuurlijk geen weet van en halen ieder voor zich de informatie van de WWW servers op. Om dit proces ietwat te stroomlijnen is het mogelijk om één of meer *proxy servers* te installeren. Dit zijn speciale server applicaties die alle HTTP requests van een bepaald (deel)netwerk op zich afgeschoten krijgen (de browsers moeten wel expliciet worden geconfigureerd om van de proxy server gebruik te maken). HTTP proxy servers houden een cache bij van populaire documenten zodat die zonder Internet verkeer te veroorzaken door de lokale gebruikers kunnen worden geraadpleegd.

Zodra hij een request ontvangt kijkt de proxy server of hij al een lokale kopie van het gevraagde document bezit. Is dit het geval dan retourneert hij deze lokale versie. Is de lokale kopie verouderd of heeft hij geen lokale kopie dan wordt het document door de proxy server bij de juiste HTTP server opgehaald. Indien deze HTTP server caching toestaat (wordt meegegeven in de HTTP response headers) maakt de proxy server gelijk een lokale kopie van het document. Als andere gebruikers het document ophalen krijgen deze vervolgens de zojuist opgehaalde lokale kopie te zien.

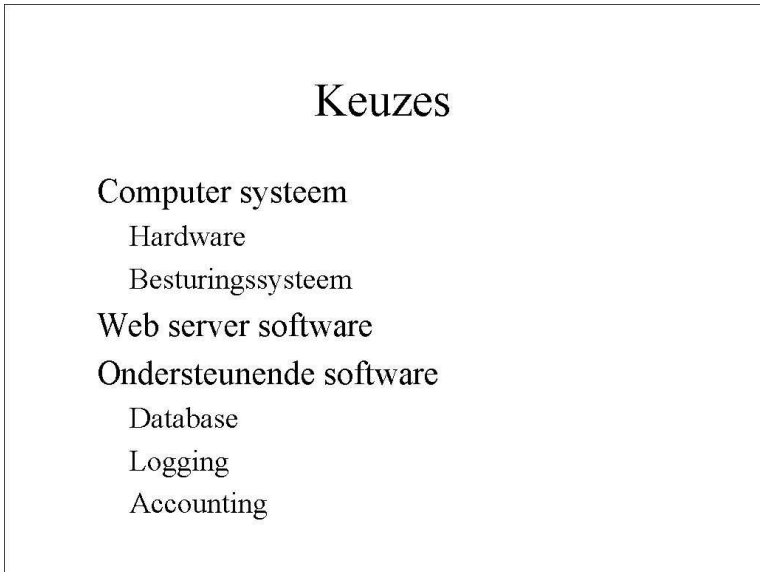
Deel II

Het opzetten van een web server

Module 8

Opzetten van een web server

8.1 Keuzes



Notities

Voordat wordt overgegaan tot de implementatie van een web server is het natuurlijk zaak dat er eerst goed wordt nagedacht over de benodigde hard- en softwarematige infrastructuur. Door de toenemende populariteit zijn er inmiddels een groot aantal mogelijkheiden om een WWW infrastructuur te realiseren. Vrijwel alle leveranciers van computersystemen, besturingssystemen, middleware (zoals databasesystemen) en applicaties claimen min of meer direkte geschiktheid voor, of koppelingen met, Internet te kunnen realiseren.

De belangrijkste keuzes vallen uiteen in de volgende categorieëen:

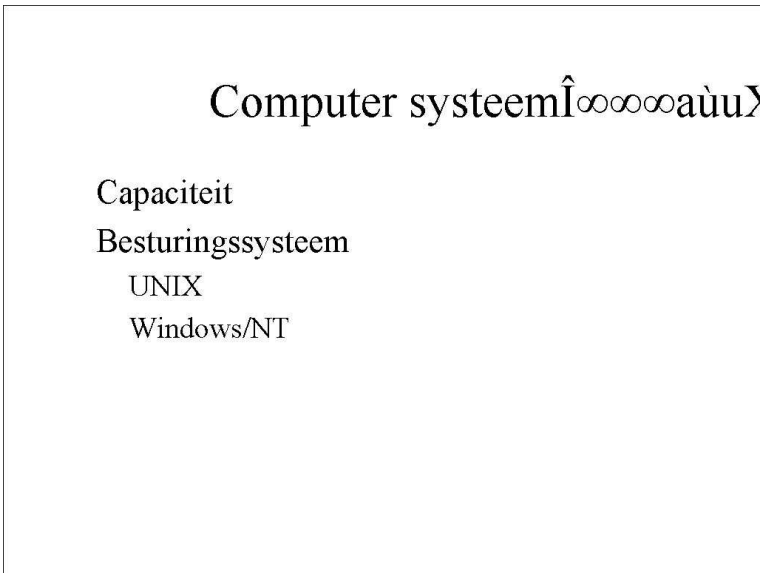
1. Computer systeem waarop de web server moet worden geïmplementeerd
2. Web server software die HTML pagina's serveert
3. Ondersteunende applicatieve en beheersoftware

De belangrijkste keuzefactoren worden gevormd door de benodigde:

- Kracht
- Performance
- Flexibiliteit

Opvallend is dat er op dit moment een groot aantal verschillende oplossingen te verkrijgen zijn voor relatief eenvoudige WWW-omgevingen. Voor het realiseren van complexe client/server applicaties op basis van WWW-technologie moet vaak nog flink wat inspanning worden geleverd om een goede infrastructuur op te zetten.

8.2 Computer systeem

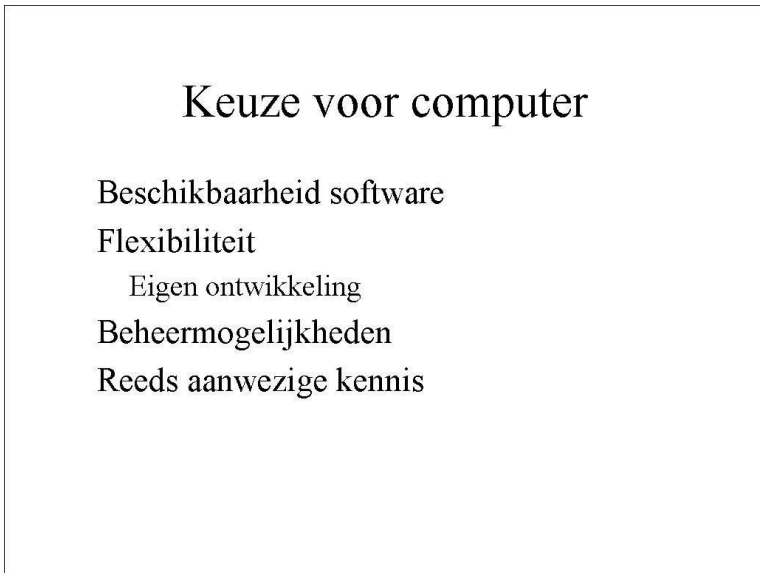


Notities

Bij de keuze voor een computer om de WWW omgeving op te realiseren moet u zich allereerst laten leiden door het gewenste operating systeem. De twee meest populaire operating systemen op dit moment voor Internet toepassingen zijn UNIX en Windows/NT. Voor beide besturingsystemen zijn momenteel ruimschoots oplossingen voorhanden om web servers mee op te bouwen. In onze ervaring geniet UNIX nog steeds de voorkeur voor complexere omgevingen vanwege de betere schaalbaarheid, grotere flexibiliteit en betere beheerbaarheid van grote UNIX-omgevingen, Microsoft doet echter zijn uiterste best om Windows/NT als aanvaardbaar server alternatief te presenteren. De grote beschikbaarheid van eenvoudig bedienbare server programmatuur en het bedieningsgemak van Windows/NT zijn pluspunten van dit O.S.

Voorts is natuurlijk de gewenste computercapaciteit van groot belang. Met de hedendaagse hardwareprijzen is vrijwel iedere omgeving tegen aanvaardbare kosten te realiseren.

8.3 Keuze voor computer



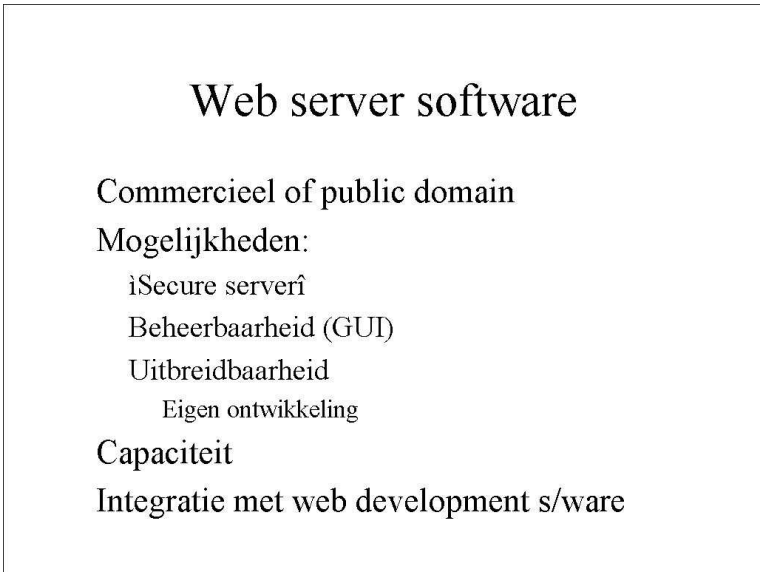
Notities

Bij de keuze voor een computer en een besturingssysteem moet u zich hoofdzakelijk laten leiden door de volgende factoren:

- Beschikbaarheid software
  Is de gewenste ontwikkel- en web server software beschikbaar op deze combinatie van hardware en O.S?
- Flexibiliteit
  In hoeverre kan op dit platform eenvoudig zelf software worden ontwikkeld?
- Beheermogelijkheden
  Kan dit platform op de gewenste wijze worden beheerd? Is er beheer op afstand (bijvoorbeeld via dial-in) mogelijk? Kan de software worden geïntegreerd met beheersoftware zoals HP OpenView, IBM Tivoli of CA UniCenter?
- Reeds aanwezige kennis
  Is er momenteel al kennis van dit platform aanwezig of moet die nog helemaal worden opgebouwd?

Met betrekking tot de capaciteit van de web server dient u zich te realiseren dat de meeste web servers tamelijk eenvoudige taken verrichten zoals het serveren van HTML documenten/componenten en het af en toe uitvoeren van serverzijde (CGI) scriptjes. De hoeveelheid transacties die op de server kan worden afgeschoten is beperkt door de snelheid van de Internet verbinding. In de meeste gevallen volstaat een relatief kleine server. De OSP Internet server bestaat bijvoorbeeld uit een 486DX66 machine met UNIX. Van capaciteitsproblemen is nog nooit sprake geweest!

8.4 Web server software



Notities

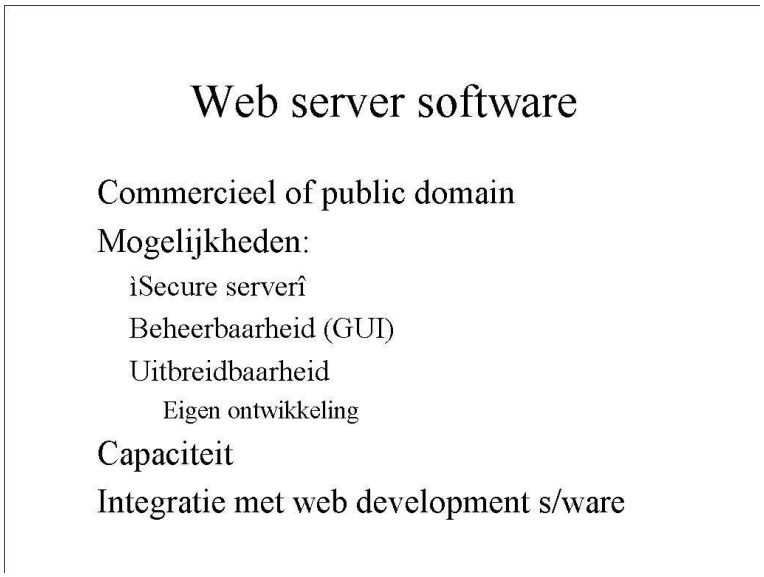
De belangrijkste software component van de web server is natuurlijk de HTTP daemon. Deze is immers verantwoordelijk voor het ophoesten van HTML documenten en het activeren van serverzijde applicaties. Alle belangrijke leveranciers leveren tegenwoordig web server software, of hebben hun applicaties aangepast om ook als web server te kunnen dienen. Een mooi voorbeeld hiervan is Lotus die met Domino een WWW-server variant van de populaire Lotus Notes groupware applicatie op de markt brengt. Dit soort web servers zijn natuurlijk niet algemeen inzetbaar, maar kunnen wel perfect worden gebruikt voor specifieke intranet/Internet toepassingen (wellicht in combinatie met NC's).

Naast de commerciële web servers zijn er ook diverse web server applicaties in het public domain. De kwaliteit van de meeste daarvan is goed tot zeer goed te noemen. Veel bedrijven gebruiken dan ook dit soort gratis web servers. Een nadeel van de meeste public domain web servers is dat ze meestal minder gebruikersvriendelijke beheerinterfaces hebben (veelal configureren met tekst configuratie bestanden)[1]. Een aantal leveranciers van besturingssystemen leveren een web server met hun O.S. mee (bijvoorbeeld de Internet Information Server in Windows/NT). Prijstechnisch is dit natuurlijk zeer aantrekkelijk.

Andere punten voor de selektie van web server applicaties staan op de slide vermeld. U dient vanzelfsprekend een goed inzicht te hebben in wat u met de web server wilt gaan doen: alleen statische pagina's serveren of uitgebreide client/server interacties aangaan. De beschikbare web server applicaties verschillen namelijk nogal in hun mogelijkheden om server zijde applicaties te integreren.

[1] vi rules!

8.5 Belangrijkste web servers



Notities

Op de slide staan de drie belangrijkste web server applicaties van dit moment genoemd. Volgens tellingen is Apache momenteel de meestgebruikte WWW server applicatie van dit moment. Professionele omgevingen die uitgebreide mogelijkheden nodig hebben zoals veilige transactieverwerking en koppelingen met databases kiezen meestal voor de Netscape server reeks of (in iets mindere mate) voor Microsoft Internet Information Server (op Windows/NT systemen).

8.6 Database systemen

Database systemen

Alleen nodig voor complexe www applicaties

Gekoppeld met transactie monitor?

Direkte Internet koppeling

- Ondervraagbaar via HTTP?
- Genereren van HTML?

Internet gateways

- In samenwerking met web server applicatie

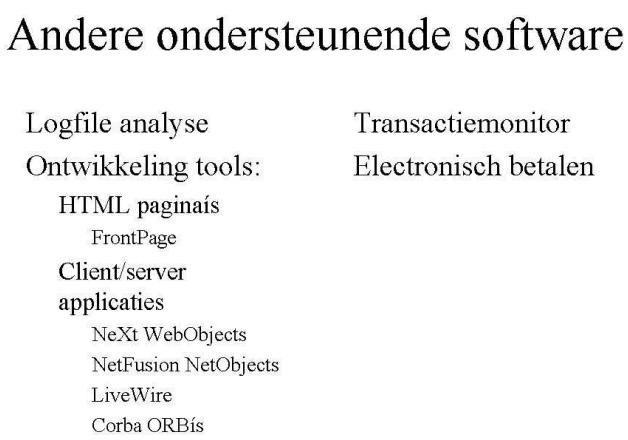
Notities

In complexere WWW omgevingen worden vaak koppelingen met databases gerealiseerd voor het opvragen van produktinformatie of het opslaan van aangeboden transacties. Voor het realiseren van een goed geïntegreerde infrastructuur is het van belang om de database server applicatie bij de overwegingen te betrekken. Een aantal database systemen kent namelijk goede/direkte koppelingen met Internet of met web server applicaties.

Zo kent SQL Server bijvoorbeeld mogelijkheden om rechtstreeks uit database tabellen HTML pagina's te genereren op basis van templates. Door triggers op de tabellen kunnen automatisch nieuwe pagina's worden gegenereerd op het moment dat er iets in de tabel wijzigt. Ook Oracle heeft het nodige aan Internet functionaliteit in zijn database systeem geïmplementeerd.

Ook zijn diverse Internet/database koppelingen gerealiseerd via speciale gateways. Dergelijke gateways vormen een HTTP request om tot een database query en genereren op basis van de uitvoer van de query een HTML antwoord. Voorbeelden van dergelijke gateways zijn de Internet Database Connector (IDC) van Microsoft (IIS/ODBC koppeling), DB2WWW (voor IBM's DB/2 database) en pg95WWW (koppeling tussen Postgres95 en Internet).

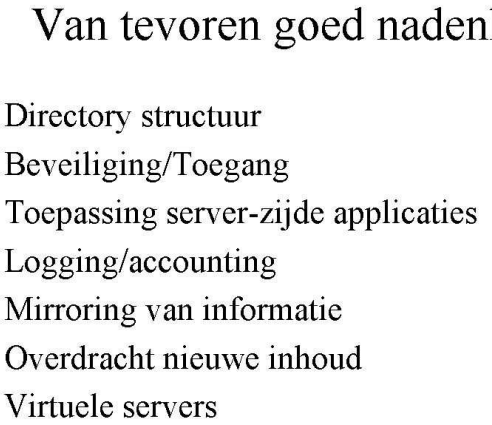
8.7 Andere ondersteunende software



Notities

Naast het O.S., de web server en de database is er vaak nog andere software nodig om de omgeving “af te maken”. Denk hierbij aan hulpmiddelen voor WWW logfile analyse (later meer hierover), ontwikkeltools voor HTML of complete web applicaties, software voor electronische betalingen of transactiemonitoren om WWW client/server applicaties veilig over meerdere systemen te verdelen.

8.8 Van tevoren goed nadenken



Notities

Vanzelfsprekend moet er voordat wordt overgegaan tot de inrichting van de WWW omgeving goed worden nagedacht over de totale inrichting. Vaak zien we dat (onder commerciële druk) WWW omgevingen zeer snel in produktie worden gebracht. In de meeste gevallen gaat dit ten koste van een zorgvuldige implementatie van de omgeving. Onderdelen waarover moet worden nagedacht zijn:

- Directory structuur
  De HTML document directorystructuur. Bijvoorbeeld aparte subdirectories voor verschillende afdelingen/applicaties/file typen. Met name het scheiden van de executables van de kale HTML kan nooit kwaad (in verband met beveiliging).

- Beveiliging en toegang tot de server
  Wie mag wat zien/doen? Hoe onderscheiden we geautoriseerde gebruikers van ongeautoriseerde gebruikers? Wie voert de toegangsbeveiliging? Vindt er auditing plaats op de web server? Met welke rechten (user) draait de web server applicatie? Gaan we gebruik maken van encryptie (SSL)?

- Toepassing server-zijde applicaties
  Wordt er gebruik gemaakt van server-zijde applicaties? Waarin en waarmee worden die ontwikkeld? Hoe wordt 'state' vastgehouden?

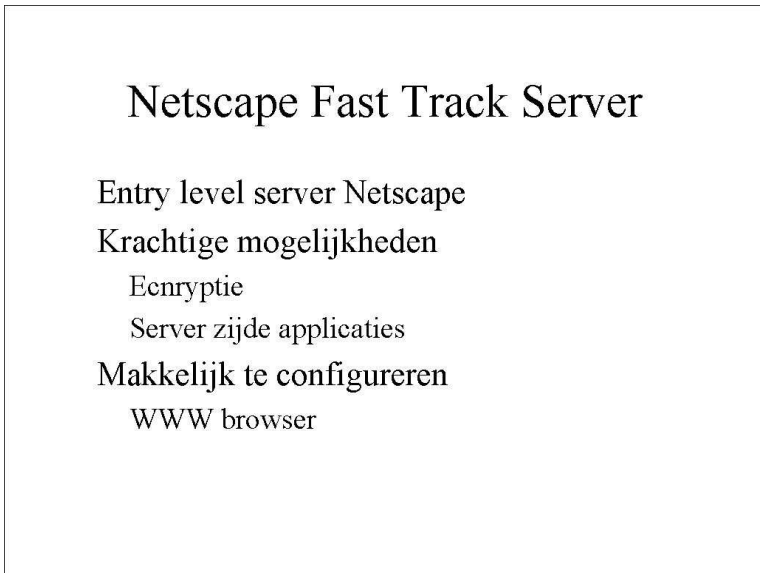
- Logging en accounting
  Worden er overzichten gemaakt van wie wat ophaalt van de server? Zo ja, is hiervoor speciale software nodig? Worden logfiles regelmatig geschoond?

- Mirroring van informatie
  Zijn er fallback web servers ingeval van storingen? Wordt informtie van anderen op deze server gemirrored?
- Overdracht nieuwe inhoud
  Hoe wordt nieuwe content overgedragen naar de produktie web server? Is er een ontwikkel/test/acceptatie omgeving voor HTML en web applicaties?
- Virtuele servers
  Gaan we meerdere virtuele servers draaienop één systeem? Zo ja, wordt er onderscheiden in IP-adressen of in TCP-poorten?

Module 9

Installatie Netscape Fast Track Server

9.1 Netscape Fast Track Server



Notities

Ter illustratie van een aantal van de eerder besproken punten hebben we in deze module de installatie en basisconfiguratie van de Netscape Fast Track Server uitgewerkt. Dit is een entry level web server van Netscape waarmee snel en eenvoudig een krachtige web server kan worden opgezet. Vrijwel alle belangrijke features (zoals encryptie en basis mogelijkheden voor serverzijde applicaties) zitten in deze server opgenomen. Alhoewel het niet de bedoeling van deze mini-cursus is om één specifiek produkt aan te bevelen illustreert de Fast Track server vrij goed welke zaken er aan de orde komen bij de installatie van een web server. De installatie en configuratie van andere servers is anders, maar wel overeenkomstig.

Een aardige feature van Netscape servers is dat de configuratie van de servers geschiedt via een “gewone” web browser. Met de eerste Netscape server (WWW, news, mail, etc.) die u installeert wordt een speciale “Admin Server” geïnstalleerd. Door met een web browser contact te zoeken met deze admin server kunt u alle andere Netscape server software installeren en configureren.

9.2 Installatie

llatie

Notities

De installatie begint met het uitvoeren van het “ns-setup” programma. Dit leidt ons eerst door een aantal stappen waarmee de software voor de admin server en de Fast Track Web Server op het systeem wordt gezet. Vervolgens start ns-setup de admin server waarna we de installatie en configuratie voortzetten via een web browser.

9.3 Installatie (cont.)

e (cont.)

Notities

HP-UX gebruikers opgelet! Netscape kent standaard niet het strakke onderscheid wat HP-UX 10 kent voor configuratie bestanden (/etc/opt), programmabestanden (/opt) en data (/var/opt). De directory die u hier aangeeft bevat standaard alle bestanden van de Netscape server. Natuurlijk kan dit met een beetje knutselen wel worden rechtgebreid.

9.4 Installatie (cont.)

e (cont.)

Notities

Na het uitpakken van de bestanden krijgen we de keuze om de nieuwe server te configureren. Let op! Het gaat hier om de admin server waarmee de echte WWW server kan worden geconfigureerd en geïnstalleerd.

9.5 Installatie (cont.)

e (cont.)

Notities

Zoals u ziet wordt er hier al met hostnamen gewerkt. Om een goede en consistente naamgeving mogelijk te maken is het van groot belang dat er in de WWW omgeving gebruik wordt gemaakt van een goede Domain Name Server.

9.6 Installatie (cont.)

e (cont.)

Notities

De Netscape admin server is een “gewone” WWW server die echter luistert op een andere TCP-poort dan de standaard poort 80. De poort waaraan de admin server wordt gebonden kan hier worden ingegeven. De rest van de configuratie van de Netscape server kan dan geschieden door met een web browser (HTML 2.0 en JavaScript compatible) te verbinden met de URL:

`http://my.server.nl:6666`

9.7 Installatie (cont.)

Notities

De admin server moet natuurlijk ook met een bepaald userid draaien. Het is verreweg het gemakkelijkst als dit de user “root” is, omdat de admin server dan ook zonder problemen alle andere Netscape servers kan stoppen en starten. Uit beveiligingsoverwegingen zouden we echter nog wel eens voor een ander userid kunnen kiezen.

9.8 Installatie (cont.)

e (cont.)

Notities

Om te voorkomen dat iedereen zonder meer contact kan maken met de admin server wordt deze beschermd met een userid en een wachtwoord. Bij het verbinden aan de admin server moet het hier gekozen userid en wachtwoord worden ingevoerd. Dit mechanisme werkt via het standaard HTTP “Basic-Authentication” mechanisme (wordt later toegelicht).

9.9 Installatie (cont.)

Notities

Als verdere beveiliging kan de admin server ook nog worden geconfigureerd om alleen connecties vanaf bepaalde systemen te honoreren. Op die manier is er een tweetraps beveiliging: de beheerder moet vanaf bepaalde systemen werken *en* hij/zij moet het admin userid en password kennen.

9.10 Installatie (cont.)

e (cont.)

Notities

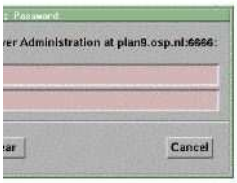
Tot slot van de configuratie van de admin server schrijft ns-setup de configuratie bestanden en een stop- en start script van de admin server naar het file systeem.

9.11 Installatie (cont.)

Notities

De rest van de installatie geschiedt via een web browser. Ns-setup biedt hier aan om deze voor ons te starten. Het mag geen verwondering wekken dat ns-setup een lichte voorkeur heeft voor Netscape Navigator als de te gebruiken web browser. Onze ervaring is dat Netscape servers ook met de Microsoft Internet Explorer kunnen worden beheerd. Echter door fouten in de Microsoft JavaScript implementatie schieten er nogal eens JScript errors door het beeld.

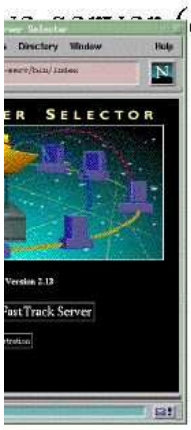
9.12 Installatie nieuwe server



Notities

Zodra u verbinding maakt met “http://my.server.nl:6666” vraagt de admin server u om het admin userid en password door middel van de getoonde dialoog box.

9.13 Installatie nieuwe server (cont.)



Notities

Na succesvol te zijn aangelogd aan de admin server komen we in verbinding met de admin server. Deze “ziet” dat er nog geen web servers zijn opgezet, en de enige keuze die hij ons biedt is dan ook het installeren van een nieuwe Fast Track Web Server.

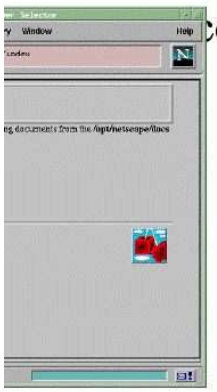
9.14 Installatie nieuwe server (cont.)

cont.)

Notities

Door voor deze installatie optie te kiezen komen we in een invulscherm waarin we de eigenschappen van de web server kunnen ingeven. Alle Netscape servers maken het mogelijk om meerdere virtuele servers op één machine te draaien. Deze servers kunnen verschillen in het virtuele IP-adres en/of in de gebruikte TCP-poort. Deze virtuele servers hebben volledig gescheiden configuraties. Op deze manier kan vrij handig een ontwikkel/test/acceptatie omgeving worden opgezet op een en dezelfde server.

9.15 Installatie nieuwe server(cont.)



cont.)

Notities

Hoera! De nieuwe server is geïnstalleerd.

Module 10

Configuratie en beheer

10.1 Configuratie

Notities

Als we terug gaan naar het hoofdmenu van de admin server (ook wel de Server Selector genoemd) kunnen we nog een server installeren of de bestaande server (hier onder de naam plan9) stoppen, starten of configureren.

In de volgende slides zullen we kort de verschillende onderdelen van de Fast Track Web Server configuratie aan de orde laten komen.

10.2 Configuratie (cont.)

Notities

De configuratieschermen van de Netscape Fast Track Web Server bestaan uit categorieën (aan de bovenzijde van het scherm), groepen instellingen per categorie (aan de linkerzijde van het scherm) en de daadwerkelijke configuratieëlementen (in het hoofdframe rechts).o

De “System Settings” categorie bevat algemene instellingen welke zijn gerelateerd aan de netwerkconfiguratie en de performance van de server.

10.3 Configuratie (cont.)

Notities

Alle web servers bieden mogelijkheden om toegang te verlenen op basis van een userid en wachtwoord (meer hierover in het beveiligingsgedeelte van de mini-cursus). Een voorbeeld hiervan is de toegang tot de admin server met het admin userid en wachtwoord. Via de getoonde configuratieschermen van de Netscape Fast Track Server kan de gebruikersdatabase van de server worden onderhouden. Ook kan de toegang van de gebruikers tot delen van de server worden ingesteld.

10.4 Configuratie (cont.)

Notities

Met de instellingen in de “Encryption” categorie kunnen instellingen rondom de Secure Socket Layer (SSL) worden geconfigureerd. Met SSL wordt de verbinding tussen de web browser en de web server beschermd met geavanceerde authenticatie en encyptietechnieken. U krijgt meer over SSL te horen in de sectie van de cursus die over WWW beveiliging gaat.

10.5 Configuratie (cont.)

Notities

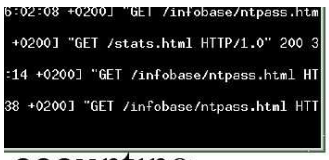
Steeds meer organisaties gaan over tot het aanbieden van complete applicaties via het World Wide Web. De kern van deze applicaties worden veelal gevormd door programma's op de server die door HTTP requests worden ge"triggered". Echter, dergelijke programma's kunnen ook beveiligingsrisico's met zich meebrengen. Om die reden is het mogelijk om het uitvoeren van programma's door de server te beperken (eventueel per directory).

10.6 Configuratie (cont.)

Notities

Web servers genereren meestal vrij uitgebreide logging over de interactie met web browsers. Zo wordt bijvoorbeeld vastgelegd welke clients welke documenten wanneer ophaalden, en welke requests (om welke reden) niet werden gehonoreerd. De Netscape Fast Track server maakt het mogelijk om deze logging via het configuratie interface op te halen en te bekijken.

10.7 Logging/Accounting



ccounting

Notities

Een voorbeeldje van de hiervoor besproken logging ziet u op de slide. Sommige web servers (zoals de Microsoft Internet Explorer) maken het mogelijk om rechtstreeks naar een database tabel te loggen, zodat met SQL queries intelligente informatie uit die logging kan worden verkregen. Tekstlogging moet meestal door een aparte tool worden omgevormd tot zinvolle rapportage. Het formaat van de tekstlogging is de facto gestandaardiseerd doordat vrijwel iedereen zich conformeert aan het log formaat van de NCSA (public domain) web server.

In de getoonde log vindt u de volgende kolommen:

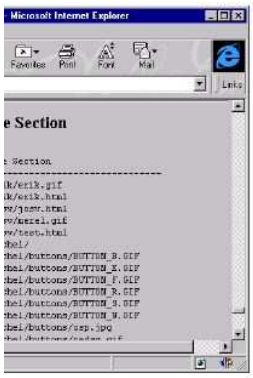
1. De naam van de host (of het IP-adres) waarvan het request kwam.
2. De datum/tijd van het request.
3. De tekst van het HTTP request (bevat commando en URI).
4. De return code van het request.
5. Het aantal bytes in het antwoord op de request.

10.8 WWW statistics

Notities

Een voorbeeld van een hulpmiddel om de web server tekst logging om te vormen tot een rapport over het gebruik van de web server is “WWWstat”. WWWstat is een Perl programma wat de tekst logfile verwerkt en een rapport met statistieken omtrent het web server gebruik genereert. Het formaat van het gegenereerde rapport is HTML. De output van WWWstat moet dus met een web browser worden bekeken. WWWstat kan prima als CGI programma draaien om “on-the-fly” WWW statistieken te genereren.

10.9 WWW statistics (cont.)



Notities

Op de slide ziet u een wat interessanter gedeelte van de output van WWWstat: de toegangsstatistieken per document (URI). Er zijn inmiddels een groot aantal verschillende applicaties om tekstuele en grafische rapportage van het WWW gebruik te genereren.

10.10 Error log

r log

Notities

Er gaat natuurlijk ook nog wel eens iets fout. De error log van de web server bevat een overzicht van mislukte HTTP requests. De meest voorkomende redenen voor het foutlopen van een request zijn het opvragen van niet bestaande documenten en het halverwege verbreken van de netwerkverbinding. Zoals uit de log kan worden gezien gebeurt het nog wel eens dat iemand het niet bestaande bestand "*robots.txt*" ophaalt. Robots.txt is een configuratiebestand voor de alomtegenwoordige Internet "search engines" (zoals Yahoo, Alta-Vista en Lycos) die web servers afstruinen en indexeren. Door specificaties in robots.txt kunnen gedeelten van de web server worden afgesloten voor alle, of met name genoemde, zoekrobots.

Deel III

WWW Beveiliging

10.11 Waarom beveiliging

Waarom beveiliging

WWW servers zijn algemeen toegankelijk

Public Relations

Commerciele toepassingen

- Financiele transacties
- Betaalde informatie

Gebruikers willen geen ongewenste bijeffecten van WWW gebruik

Notities

Het ligt natuurlijk volledig voor de hand waarom WWW omgevingen goed moeten zijn beveiligd. Het gaat hier immers doorgaans om publiek toegankelijke servers die vitale functies voor een organisatie verrichten. Zelfs indien een WWW server geen financiële transacties verwerkt kan de public relations schade ten gevolge van het “hacken” van de site nog enorm zijn. Het Amerikaanse ministerie van justitie ondervond dit tot toen haar web site eind vorig jaar door een aantal inbrekers onherkenbaar werd gewijzigd.

Beveiliging werkt echter twee kanten op: organisaties willen natuurlijk ten koste van bijna alles voorkomen dan hun sites worden gekraakt, gebruikers willen daarentegen door het “surfen” geen virussen of andere ongewenste programma’s binnenhalen.

10.12 Beveiligingsproblemen

Beveiligingsproblemen

WWW gebaseerd op open protocollen

Doel:

- wel: informatieverspreiding
- niet: veilig interactief transactiemedium

Problemen:

- Privacy
- Integriteit
- Authenticatie
- Beveiligingsgaten

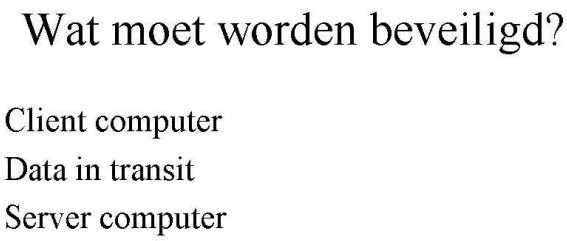
Notities

Allereerst moet natuurlijk worden opgemerkt dat het World Wide Web (net als Internet) nooit verzonnen is voor het gebruik wat er heden ten dage van wordt gemaakt. Het WWW is indertijd opgezet voor het vrijelijk verspreiden van informatie en het algemeen toegankelijk maken van documenten. Dientengevolge speelde beveiligingsaspecten geen enkele rol bij het bedenken van de standaarden die in het web worden toegepast.

Om deze reden zijn de protocollen en standaarden die algemeen in het web worden gebruikt (zoals HTTP en HTML) in zijn algemeen niet uitgerust met beveiligingsgerelateerde mogelijkheden. Het is echter wel opvallend te noemen dat ook bij modernere uitbreidingen aan het web beveiliging niet hoog in het vaandel lijkt te hebben gestaan. Er is met name gedacht aan het leveren van flitsende functionaliteit, en minder aan het beveiligen van de bestaande en nieuwe features. Dientengevolge doen alle traditionele beveiligingsproblemen zich op het web voor:

1. Privacy
   De verzonden data is niet beveiligd tegen afluisteren.
2. Integriteit
   Er kan niet met zekerheid worden gesteld of de data niet tussentijds is gewijzigd (in de verzending).
3. Authenticatie en Autorisatie
   Wie zijn de client en de server, en mogen ze wat ze willen?

10.13 Wat moet worden beveiligd?



Notities

De volgende elementen in het web moeten dan ook afdoende worden beveiligd:

1. De client
2. De server
3. De data in transit

10.14 Client computer

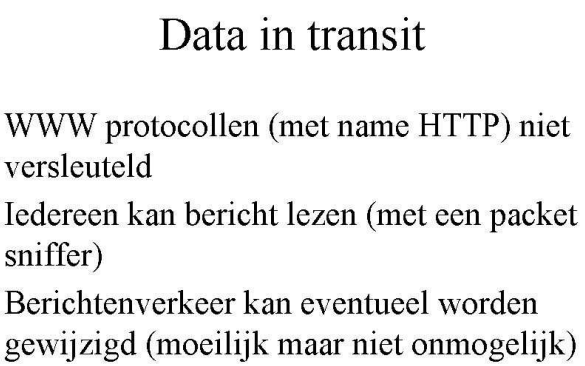
Notities

Het grote gevaar voor de World Wide Web client is het direkt of indirekt binnenhalen en (doen) uitvoeren van programmatuur die allerlei ongewenste verwerkingen beginnen. Kortom: virii[1]. Helaas is het gevaar daartoe alleen maar toegenomen. Voor een flitsende WWW presentatie is het wenselijk om lokaal allerlei programma's uit te laten voeren: bijvoorbeeld voor animaties, niet-standaard GUI controls of het presenteren van complexe data zoals PostScript, PDF of VRML. Dit betekent dat de WWW browser moet worden uitgebreid met "vreemde" programmatuur om die mogelijkheden te kunnen bieden. Hiermee wordt het gevaar voor virii natuurlijk aanzienlijk vergroot.

Daarnaast bestaat het gevaar van de ongewenste vrijgave van allerlei gegevens. WWW browsers sturen meer informatie met een request mee dan je wellicht zou denken. Op die manier vindt het web surfen steeds minder anoniem plaats dan je wellicht zou denken. Als laatste dient nog te worden genoemd dat vele browsers bugs bevatten die de WWW client in gevaar (kunnen) brengen.

[1] Meervoud van het bekende latijnse zelfstandige naamwoord "virus".

10.15 Data in transit



Notities

Het standaard World Wide Web protocol voor het verzenden en ophalen van gegevens (HTTP) bevat geen mogelijkheden ter beveiliging van de datastroom. Dit betekent dat een ieder die toegang heeft tot de tussenliggende netwerkverbinding (bijvoorbeeld alle medewerkers van de Internet Access Provider) de data kan bekijken en eventueel kan wijzigen. Dit laatste is niet gemakkelijk, maar wel mogelijk. Ook kent HTTP geen mogelijkheden om te weten wie de afzender is van de gegevens.

10.16 Server computer

Server computer

Algemeen toegankelijk (via Internet)

Algemene Internet server beveiliging

Firewalls, O.S. beveiliging e.d.

Toegang tot de WWW server

Alleen geautoriseerde gebruikers?

Complexe WWW server applicaties

Worden aangestuurd van ìbuitenî

Notities

En tenslotte dient ook de WWW server computer afdoende te worden beveiligd. Servers zijn altijd aantrekkelijke targets voor inbrekers: ze zijn meestal 7x24 uur online, algemeen toegankelijk, en bieden meestal (zodra gekraakt) toegang tot weer andere computers om te kraken. Een extra moeilijkheid in het geval van het World Wide Web is dat WWW servers requests uitvoeren die volledig van buiten af worden geparametriseerd. Bij bugs in de WWW applicaties kunnen zij wellicht worden verleidt om andere dingen te doen als dat de maker zich had voorgesteld. Dit betekent dat het voor organisaties aantrekkelijk kan zijn om de WWW server alleen open te stellen voor geauthenticeerd en geautoriseerde gebruikers. Helaas betekent dit in veel gevallen dat de beheerder van de site allerlei extra maatregelen moet nemen.

Module 11

WWW Client Beveiliging

11.1 Bugs

Bugs

Enorme ontwikkelsnelheid van browser leveranciers

Opname nieuwe/complexe functionaliteit

Diverse beveiligingsgerelateerde bugs in Netscape Navigator en Microsoft Internet Explorere

Notities

Een groot gevaar voor WWW gebruikers wordt gevormd door de bugs die met de regelmaat van de klok opduiken in de moderne WWW browsers. De concurrentiestrijd tussen de twee belangrijkste leveranciers is zo groot dat produkten voordat ze goed en wel getest zijn op de markt worden gegooid. Het is zelfs zo erg dat alle klanten uitgebreid de beschikking krijgen over de beta versies om de leverancier te helpen bij het oplossen van de problemen[1]. Er wordt door de leveranciers zeer hard gewerkt aan het uitbreiden van de functionaliteit, met als gevolg dat de kwaliteit en veiligheid van de code nogal eens achterblijft. Als gevolg hiervan worden zelfs in de “release” versies van de browsers nogal eens beveiligingsproblemen aangetroffen die door snoodaards kunnen worden misbruikt om uw systeem te corrumperen. Dit heeft er ook toe geleidt dat gebruikers actief de beveiligings “patches” van de leveranciers moeten volgen en aanbrengen.

[1] Zou u een beta van uw huidige automerk gaan rijden op het gevaar af dat het stuur er plotseling afvalt?

11.2 Active Content

Active Content

Uitvoeren van programmaís op de client:
- Helper
- Plug-in
- Java
- JavaScript / VBScript

Gevaar:
- Uitvoer ìgedownloadî programma
- Lokale verwerking ìgedownloadeî data

Notities

Het grootste gevaar voor de gebruikers wordt gevormd door alle vormen van “Active Content”: de mogelijkheden die het web tegenwoordig biedt om verwerking en presentatie van complexe data te verplaatsen naar de client. De diverse mogelijkheden daartoe hebben zo ieder hun beveiligingsaspecten. De algemene gevaren zijn:

1. Een programma (met onbekende inhoud) wordt gedownload en lokaal uitgevoerd (Java, JavaScript, ActiveX, VBScript).
2. Een complexe datafile (bevattende data en wellicht ook commando’s) wordt gedownload en lokaal geïnterpreteerd door een krachtige interpreter (helpers en plug-ins).

11.3 Helper applicaties en plug-ins

Helper applicaties en plug-ins

Complexe data

Ingesloten commandoís:

- Microsoft Word
- PostScript
  - Oplossing: gebruik *GhostView* auto-safe mode
- MacroMedia ShockWave

Voorzichtigheid is geboden!

Notities

Het op één na grootste gevaar van dit moment[2] wordt gevormd door helper applicaties en plug-ins die complexe databestanden die door de web server worden overgezonden lokaal interpreteren. Onder deze categorie vallen “office” applicaties (zoals tekstverwerkers, spread-sheets en dergelijke), maar ook de interpreters voor PostScript en multi-media bestanden zoals VRML[3]. De “feature drive” van de laatste jaren heeft ervoor gezorgd dat die applicaties allerlei mogelijkheden kennen voor het opnemen van commando’s, macro’s en andere ongewenste mogelijkheden. Zo kent Microsoft Word bijvoorbeeld de befaamde Word macro’s waarmee van alles en nog wat kan worden geprogrammeerd. Vervolgens kent Word de mogelijk om een “auto execute” macro in een document te specifieren, die automatisch moet worden uitgevoerd zodra het document wordt geopend. Het ophalen en automatisch openen van een Word document is dus levensgevaarlijk! Ook andere file formaten (applicaties) kennen dergelijke mogelijkheden. PostScript kent bijvoorbeeld mogelijkheden voor het manipuleren van bestanden.

De oplossing hiervoor is natuurlijk om alleen veilige interpreters te gebruiken (of bestaande interpreters veilig te configureren). Zo kent de public domain PostScript interpreter “GhostView” een veilige modus (“safe mode”) waarin ingebakken commando’s in de PostScript stream niet worden uitgevoerd.

[2] Ra, ra, wat is het grootste gevaar?
[3] Virtual Reality Modelling Language

11.4 Java

Java

Beveiliging onderdeel Java taal specificatie
Sandbox model
Conceptueel in orde
Bugs in de implementatie
Java *policy* instellingen in browser
Nieuw: digitaal signeren van Java applets
Wel: *ìDenial of serviceî* en *ìspoofingî* aanvallen

Notities

Aan de taal Java is de laatste tijd enorm veel aandacht besteed. Dit komt niet in de laatste plaats omdat de Java taal en omgeving standaard zijn voorzien van een groot aantal eigenschappen die het transporteren en veilig lokaal uitvoeren van Java programma's mogelijk maakt. Java programma's draaien in een interpreter die iedere aktie van het Java programma kan controleren. Java "applets" die in de context van een World Wide Web browser draaien wordt het recht ontzegd om acties uit te voeren die potentieel onveilig zouden kunnen zijn. Zo kunnen Java applets geen bestanden manipuleren op de lokale hard disk en ook geen systeemcommando's geven.

Het Java "sandbox" model is conceptueel in orde. In theorie beveiligt het de WWW client tegen iedere ongewenste intimiteit. Tegen bugs in de Java implementatie is natuurlijk niemand bestand. Een aantal van de Java beveiligingsproblemen die in de begintijd van Java zijn gemeld waren dan ook geen problemen van Java, maar van de Java implementatie in de browser. Wel kan Java worden misbruikt voor het monopoliseren van systeemhulpbronnen (met name CPU en geheugen, de zogenaamde "Denial of Service" aanvallen) en het weergeven van informatie die slecht is voor de menselijke ziel. Ook kan Java worden misbruikt om dialoogboxen te genereren die door de gebruiker zouden kunnen worden misbegrepen als authentieke dialoogboxen van het besturingssysteem. Op die manier zou de gebruiker kunnen worden verleid om bijvoorbeeld wachtwoorden in te geven welke door de Java applet zouden kunnen worden doorgespeeld aan de web server. We noemen dergelijke aanvallen ook wel "spoofing" (neppen).

Een nadeel van Java applets is natuurlijk dat ze door de ingebouwde beveiliging ernstig zijn gelimiteerd in wat ze kunnen. Zo behoort het afdrukken van informatie op de

printer en het opslaan van gegevens op de lokale hard disk niet tot de mogelijkheden. Er zijn ontwikkelingen in gang gezet om Java applets digitaal te signeren door de auteur, om vervolgens met browser instellingen bepaalde mogelijkheden open te zetten voor applets die door vertrouwde software auteurs zijn gesigneerd. In theorie kunnen dergelijke autorisaties zeer fijnmazig zijn. Doordat de Java bytecode interpreter de volledige controle heeft over de applet kan iedere actie in theorie afzonderlijk worden geautoriseerd.

11.5 JavaScript en VBScript

JavaScript / VBScript

Zeer beperkte talen

Geen toegang tot client computer resources

Wel:

ìDenial of serviceî aanvallen

ìSpoofingî aanvallen

Netscape werkt aan uitbreidingen!

Meer toegang op basis van digitale handtekeningen

Notities

De WWW script talen JavaScript en VBScript vormen wederom een aparte categorie. Het betreft hier programma's die als onderdeel van de HTML pagina worden meegezonden en lokaal worden geïnterpreteerd door de WWW browser. Doordat deze script programmeertalen slechts zeer beperkte mogelijkheden bieden kunnen zij niet direct worden gebruikt om de client te manipuleren. Wel zijn ook hier de "Denial of Service" en "spoofing" aanvallen mogelijk. Ook voor JavaScript geldt dat wordt gewerkt aan uitbreidingen die JavaScript scripts meer toegang tot de client geven op basis van digitale handtekeningen.

11.6 ActiveX

ActiveX

Downloaden machinetaal en uitvoeren!
ActiveX controls digitaal gesigneerd.
AuthentiCode technologie
ìInternet Exploderî

Notities

Een werkelijk beveiligingsdrama wordt gevormd door ActiveX. Het betreft hier Microsoft's technologie voor het ophalen, installeren en uitvoeren van executables door WWW browsers (alleen op WIN32 platformen door de Microsoft Internet Explorer). Het krachtige aan ActiveX is het zelf-installerende mechanisme en het feit dat de controls (meestal geschreven in C/C++) alle eigenschappen van het onderliggende besturingssysteem (Windows 95 of Windows/NT) kunnen benutten. Het grote gevaar is natuurlijk dat we geen flauw benul hebben van wat de control precies doet. Eenmaal geïnstalleerd kan het door de browser worden uitgevoerd, waarna de control zonder tussenkomst van wie dan ook zijn gang kan gaan.

Microsoft heeft getracht hier omheen een beveiligingsschil te leggen welke is gebaseerd op digitale handtekeningen. Software leveranciers kunnen een "Digital ID" kopen bij VeriSign. Met deze ID kunnen ze hun software signeren. Met het kopen van een ID beloof je dat je geen virii zult signeren[4]. Bij het installeren van de ActiveX controls krijgt de gebruiker (onder invloed van de Explorer beveiligingsinstellingen) de keuze om de control te weigeren. Deze technologie gaat door het leven als "AuthentiCode".

Een Amerikaanse programmeur stelde de proef op de som en kocht een VeriSign Digital ID. Vervolgens schreef hij een ActiveX control die het systeem van de web surfer herstartte, de "Internet Exploder". Bezoekers van zijn site (www.cybersnot.com) kregen te zien dat er een control werd geïnstalleerd welke was gesigneerd met een door VeriSign afgegeven Digital ID. Na acceptatie, ... BOEM! Na wat heen en weer gepraat trok VeriSign het Digital ID in. Helaas was Microsoft vergeten om bij het installeren van een control te verifiëren of het digitale ID waarmee de control was gesigneerd nog

[4] en zelfs criminelen komen die belofte natuurlijk na, ... *Not*!

wel geldig was. Bestaande versies van de Internet Explorer konden dus nog naar hartelus de Internet Exploder draaien. Pas in latere versies is deze ommissie rechtgetrokken. Voorstanders van ActiveX stelden dat het AuthentiCode mechanisme volgens planning had gewerkt en dat het gevaar was bezworen. Tegenstanders stelden echter (onzes inziens terecht) dat de enige reden dat het Digitale ID kon worden ingetrokken was omdat de auteur van de Explorer zo openlijk was over zijn bedoelingen. Hackers van een bekende Hamburgse “underground” ontwikkelden vervolgens een ActiveX control waarmee ongemerkt transacties werden toegevoegd aan de bestanden van een populaire “home finance” applicatie (Quicken) welke ook Tele-Banking faciliteiten in zich had.

11.7 Digitaal gesigneerde code

Digitaal gesigneerde code

ìSigned Code is Not Safe Codeî

Belangrijke beveiligingsbeslissingen komen bij de gebruiker te liggen!

Kan de *ìsignerî* de kwaliteit van de code garanderen?

Notities

In zijn algemeenheid is het natuurlijk een slecht idee om programma's uit semi-onbekende bron te downloaden en uit te voeren. Het gevaar van virii is helaas al te bekend. Toch is er in het Internet tijdperk een vernieuwd enthousiasme onder de gebruikers waarneembaar om precies dat te doen. Hele generaties Internet gebruikers zijn opgegroeid met het downloaden en uitvoeren van browsers en plug-ins.

Er is een duidelijke trend waarneembaar om dergelijke programma's digitaal te signeren, en autorisaties uit te delen op basis van de digitale handtekening. Een duidelijk voorbeeld hiervan is ActiveX's AuthentiCode, maar ook Netscape en Sun zijn bezig met het bedenken van uitbreidingen aan Java en JavaScript waar digitale handtekeningen een rol spelen bij de autorisaties van het programma op de client. Alhoewel aan het digitaal signeren van programmatuur duidelijke voordelen zitten, kleven er ook een aantal bezwaren aan.

Ten eerste geeft een digitale handtekening alleen aan dat de "signerïnstaat voor de identiteit van de auteur van het programma. Die auteur kan vervolgens best een crimineel zijn. Het is voor de instelling die de digitale handtekening zet natuurlijk ondoenlijk om de code van het getekende programma te controleren. Het voorbeeld van de Internet Exploder laat zien dat iemand met een digitaal ID best een vijandig programma kan schrijven.

Een ander nadeel van digitale handtekeningen is dat in een aantal gevallen de beslissing om een niet-getekend programma (of een programma waarvan de handtekening niet klopt) toch uit te voeren. Dit legt een belangrijke beveiligingsbeslissing in de handen van de gebruiker! In onze optiek is de gemiddelde gebruiker niet in staat om dit soort beslissingen te nemen.

11.8 Ongewenste vrijgave

Ongewenste vrijgave

Browsers sturen veel informatie mee in HTTP request

ìReferî link

Misbruik: Opheffen anonimiteit

Notities

Het laatste “gevaar” voor de gebruiker is het ongewenst vrijgeven van informatie. Het originele websurfen was behoorlijk anoniem, WWW servers konden doorgaans niet aan een HTTP request zien van wie het request afkomstig was. Echter, om servers in staat te stellen om meer en beter geformatteerde informatie beschikbaar te stellen sturen browsers steeds meer informatie mee over wie de gebruiker is, welke browser hij/zij gebruikt, hoe groot het scherm is en hoeveel kleuren het scherm ondersteunt. Servers gebruiken die informatie om aangepaste HTML-pagina’s en plaatjes op te sturen. Steeds vaker wordt echter die informatie gebruikt om marketing doeleinden na te streven.

Zo kan een server bijvoorbeeld “cookies” uitdelen aan bezoekende browsers. Iedere keer als de browser de site bezoekt geeft hij weer het cookie mee. Op die manier kan een server zien dat het om een gebruiker gaat die al eens eerder is langsgeweest, en kan een coherente reclamecampagne richting de gebruiker worden gevoerd. In theorie zouden meerdere organisaties hun cookie logs kunnen consolideren om zodoende het spoor van de gebruiker over Internet te kunnen volgen.

Een ander fenomeen is de “Refer” link. Browsers sturen vaak bij het opvragen van een pagina via een hyperlink de URL mee van de verwijzende pagina (de “Referer”). Op die manier kan een site weten via welke weg een gebruiker aangekomen is, wat op zich weer interessante informatie kan zijn. Zo is er een Amerikaans marketingbedrijf wat andere organisaties betaalt om verwijzingen naar plaatjes op hun eigen site op te nemen. Door een combinatie van cookies en refer links weet de server van het marketingbedrijf welke klant het betreft en van wie de verwijzing af komt. Op die manier kan de marketing server een coherente reclamecampagne voeren zelfs als de

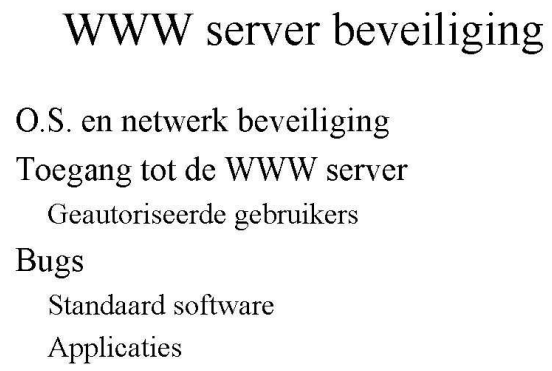
gebruiker van site naar site hopt.

Om de hierboven genoemde redenen bieden moderne browsers vaak de mogelijkheid om het accepteren van cookies tegen te gaan of onderhevig te maken aan gebruikers-toestemming. Een belangrijk nadeel hiervan is dat cookies ook gebruikt worden voor legitieme WWW applicaties.

Module 12

WWW Server Beveiliging

12.1 WWW server beveiliging



Notities

Een WWW server dient natuurlijk zeer goed te zijn beveiligd tegen ongewenste activiteiten van buitenstaanders. Net als bij alle andere Internet servers (zoals FTP servers, email servers en DNS servers) is een goede algemene bescherming (hier O.S. bescherming genoemd) uitermate belangrijk. Dit betekent onder andere dat de server alleen moet worden opengesteld voor de algemeen toegankelijke services (en bijvoorbeeld niet voor telnet). Daarnaast zijn er echter ook nog beveiligingsproblemen die gerelateerd zijn aan de WWW server functionaliteit. Zo is het bijvoorbeeld niet ongebruikelijk om een WWW server alleen open te stellen voor geautoriseerde gebruikers. Hier moet echter wel expliciet in worden voorzien door de web server of de web applicatie. Ook zijn WWW servers gevoelig voor bugs in WWW applicaties en in de WWW server software. Door een goede en robuuste implementatie kunnen we de dergelijke bugs voorkomen of de impact ervan verkleinen.

12.2 O.S. en netwerkbeveiliging

O.S. en netwerkbeveiliging

- Firewall
- Fysieke beveiliging
- Veilige toegang tot systeem
  - Voor beheerdoeleinden
- *Minimal service*
- Goed systeembeheer

Notities

De basis van de WWW server beveiliging wordt natuurlijk gevormd door de beveiliging van de computer waarop de WWW server applicatie draait. Er zijn intussen hele boeken volgeschreven over UNIX, Windows/NT en Internet beveiliging. Om die reden gaan wij op dit onderwerp hier niet al te diep in.

De toegang tot de server dient natuurlijk te zijn afgeschermd met een goede firewall. Deze dient in ieder geval de TCP poorten die voor de WWW server nodig zijn (80 en wellicht 443) door te laten. De poorten van de applicaties die niet worden gebruikt dienen door de firewall te worden tegengehouden. Vanzelfsprekend dient de toegang tot het systeem goed te zijn afgeschermd. Het verdient wellicht aanbeveling om de systeembeheertoegang af te schermen met “one-time” wachtwoorden op basis van beveiligingscalculators. Verder dient het systeem goed te worden beheerd. Dit betekent dat de logging regelmatig moet worden nagespit, er backups moeten worden gemaakt, en dat beveiligingspatches en updates op het operating systeem tijdig (maar niet te voorbarig) moeten worden aangebracht.

Voor een goed overzicht verwijzen wij gaarne naar één van de uitstekende boeken op dit onderwerp.

12.3 Toegang tot de web server

Notities

Een probleem waar veel web beheerders zich voor zien gesteld is het afsluiten van (een gedeelte van) de site voor niet-geautoriseerde gebruikers. Anders gezegd willen we het geheel of een deel van de site alleen openstellen voor bepaalde gebruikers. Meestal gaat het dan om gedeelten die alleen toegankelijk zijn voor betalende of geregistreerde klanten.

12.4 Toegangsbeveiliging

Toegangsbeveiliging

ìHidden URLî
Afzender hostnaam of IP-adres
WWW authenticatie
Applicatieve authenticatie
ìClient certificatesî (SSL, S-HTTP)

Notities

Voor het afsluiten van (delen van) sites zijn een aantal mogelijkheden. Zij verschillen met name in de mate van beveiliging die ze bieden en de mate van beheer die nodig. Vanzelfsprekend kunnen meerdere van de op de slide genoemde methoden boven op elkaar worden gestapeld om zodoende de beveiligingsgraad nog verder te verhogen. Bedenk hierbij wel dat teveel beveiligingsmaatregelen allerlei ongewenste bijeffecten kan hebben. Zo kan het gebeuren dat gebruikers de site niet meer zullen benaderen, of dat ze zelf de beveiliging gaan trachten te omzeilen om de situatie nog werkbaar te houden.

12.5 Hidden URL's

Hidden URL

HTML document / directory die niet wordt aangewezen door andere pagina

Klant moet het bestaan weten

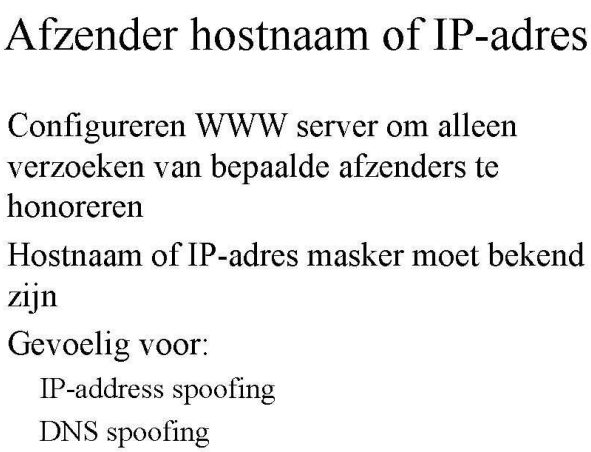
Wordt niet gevonden door zoekrobots

Slechte beveiliging

Notities

Een nog wel eens toegepaste truc is die van de verborgen URL. Hierbij wordt een HTML pagina of directory op de server opgenomen zonder dat er vanuit de "home page" direkt of indirekt naar wordt verwezen. De enige manier om die pagina dan te raadplegen is doordat je van het bestaan van de URL weet. Vanzelfsprekend is dit een niet zo beste methode van beveiliging. Als het bestaan van de URL eenmaal bekend wordt is er geen enkele toegangscontrole meer op het benaderen ervan. Waarvoor het wel heel erg handig is, is om bepaalde publieke informatie (bijvoorbeeld nieuwe site informatie) alsvast op de site te zetten en aan een kleine club gebruikers bekend te maken.

12.6 Afzender hostnaam en IP-adres



Notities

Een andere mogelijkheid om de toegang tot de WWW server te beperken is door het configureren van de web server applicatie om voor het geheel of delen van de site slechts verzoeken toe te staan vanaf bepaalde client computers (gespecificeerd door hostnamen of IP-adressen). Hierdoor kunnen bijvoorbeeld bepaalde pagina's alleen ter beschikking worden gesteld aan interne gebruikers. Alhoewel dit geen waterdichte methode is, is zij wel in bepaalde omstandigheden goed bruikbaar. Het vereist natuurlijk wel dat de hostnamen/IP-adresen van de legitieme gebruikers vooraf bekend zijn. In het geval van gebruikers die via hun ISP inbellen (met als gevolg sterk wisselende IP-adressen) werkt deze methode natuurlijk niet zo "bien". Wel dient u bij het toepassen van deze beveiliging er rekening mee te houden dat alle beveiliging op basis van afzenderadressen gevoelig is voor IP-adres spoofing en DNS spoofing. Met een goed geconfigureerde firewall kan een gedeelte van deze spoofing problematiek weer worden ondervangen.

12.7 WWW authenticatie

WWW Authenticatie

Client doet HTTP Get

Server antwoordt met

401 Not Authorized, en

îWWW-Authenticateî header in antwoord

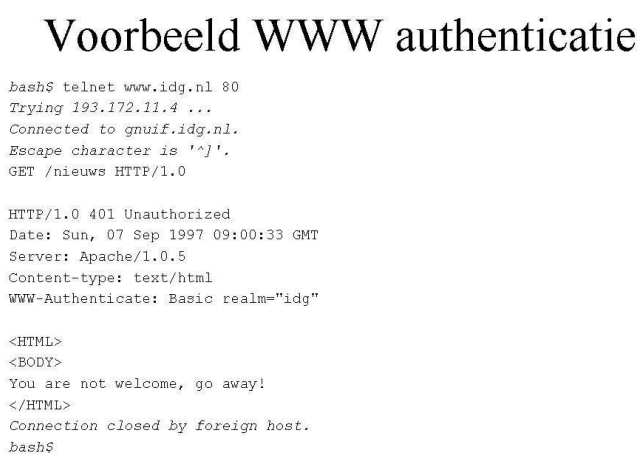
Browser vraagt userid en wachtwoord, en

Herhaalt HTTP verzoek met userid en wachtwoord informatie

Notities

Naast de reeds genoemde mogelijkheden zitten er ook in het World Wide Web basisprotocol (HTTP) features waarmee de toegang tot (een gedeelte van) de WWW site kan worden voorbehouden aan geautoriseerde gebruikers. In het geval een client een pagina opvraagt die alleen voor geautoriseerde gebruiker is bedoeld dient de browser in het request extra gegevens (headers) mee te geven waarin de authenticatie gegevens (meestal userid en wachtwoord) staan. Indien de toegang niet kan worden verleend (omdat de authenticatie gegevens niet kloppen, er geen authenticatie gegevens zijn meegestuurd of de "Access Control List" van de pagina geen toegang verleent aan deze gebruiker) antwoord de server met een foutcode "401 Not Authorized". De browser reageert dan met een dialoogscherm waarin de gebruiker nieuwe authenticatie parameters (userid en wachtwoord) kan ingeven. De browser herhaalt dan het verzoek met de nieuwe gegevens. WWW servers bieden dan ook mogelijkheden om gebruikers op te voeren en om gebruikers (groepen) toegang te verlenen tot pagina's en/of directories.

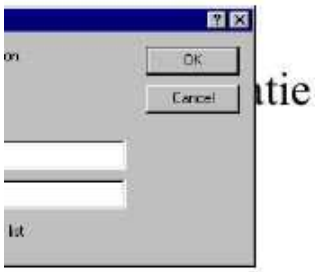
12.8 Voorbeeld WWW authenticatie



Notities

Op de slide ziet u een voorbeeld van hoe de WWW server reageert indien we een HTTP GET opdracht geven zonder de juiste authenticatie gegevens mee te sturen. Het “401” antwoord gaat vergezeld van “response headers” die gegevens bevatten omtrent het soort authenticatie wat de server kan verwerken. Het is mogelijk dat de server naast de “Basic” authenticatie ook nog andere authenticatievormen kent. Indien dit het geval is zal de server in het antwoord alle ondersteunde authenticatievormen meegeven. De browser kan dan kiezen welke authenticatiemethode hij ondersteunt. De Microsoft Internet Information Server ondersteunt bijvoorbeeld ook “NTLM” authenticatie. De Microsoft Internet Explorer begrijpt die ook en zo kunnen ze samen NTLM authenticatie uitvoeren (standaard Windows challenge response met het userid en wachtwoord waarmee de gebruiker is aangelogd aan Windows).

12.9 Voorbeeld WWW authenticatie (cont.)



Notities

De browser die een WWW “Basic” authenticatieverzoek ontvangt reageert door het tonen van een dialoog waarin de gebruiker zijn login gegevens kan inkloppen. Op de slide ziet u een voorbeeld van een dergelijke login prompt.

12.10 Applicatieve authenticatie

Applicatieve authenticatie

WWW applicatie:
- Bouwt login scherm
- Vraagt userid + wachtwoord
- Eigen gebruikers en autorisatie database

Meer fijnmazige autorisatie mogelijk

Koppeling met andere gebruikers database mogelijk
- Bijvoorbeeld O.S.

Notities

Veel applicaties kiezen er ook voor om hun eigen authenticatie te implementeren. Een voorbeeld hiervan is de OSP triviant applicatie die in een eerdere module is behandeld. De applicatie genereert zijn eigen login schermen (in HTML/Java/JavaScript of wat dies meer zij) en handelt de login poging af. Hiermee heeft de applicatie natuurlijk een veel betere controle over wie er aanlogt, en zijn er nog allerlei applicatieve autorisatieniveau's mogelijk. Ook kan er in dit geval voor worden gekozen om de gebruikersdatabase van de applicatie te integreren met een andere gebruikersdatabase (bijvoorbeeld die van het email systeem). Een belangrijk nadeel is natuurlijk dat het extra ontwikkelinspanning voor de applicatiebouwers betekent.

12.11 Voorbeeld applicatieve authenticatie

Notities

Wellicht ten overvloede hier nog een voorbeeld van hoe een applicatie zijn eigen authenticatie zou kunnen regelen. De scripts achter OSP Triviant zijn verantwoordelijk voor het afhandelen van heel de login poging.

12.12 Client certificates

Client certificates

In combinatie met SSL of S-HTTP
Digitaal identiteitsbewijs
ìGegarandeerdî door afgevende instelling
  Digitale handtekening
Distributieprobleem

Notities

De allermodernste manier van authenticatie is het over en weer presenteren van digitale certificaten. Het gaat hier om digitale identiteitsbewijzen die door een afgevende instantie worden gegarandeerd (met een digitale handtekening). In de volgende module wordt dieper op deze digitale certificaten ingegaan.

12.13 Veilige CGI/API applicaties

Veilige CGI/API applicaties

WWW server applicaties bevatten vaak bugs

Extern geparametriseerd

Aanroepen met:

- Verkeerde parameters
- Andere parameters
- Onverwachte parameters

Notities

Een belangrijk beveiligingsprobleem in WWW omgevingen wordt gevormd door de WWW serverzijde applicaties. Het betreft hier programma's in één of andere programmeertaal (meestal PERL, shell script, C/C++, Java of Visual Basic) die worden geactiveerd op basis van HTTP verzoeken. De invoerparameters van die programma's zitten in het verzoek "verstopt", en worden op een bepaalde manier doorgegeven aan de applicatie. Hierin zitten hem dan ook de meeste problemen. Door handmatig een HTTP verzoek te construeren met teveel, te weinig of onverwachte parameters (bijvoorbeeld te lang) kunnen veel applicaties worden overgehaald om verkeerd te reageren. Op die manier kunnen bijvoorbeeld andere gegevens worden overschreven of onverwachte commando's worden uitgevoerd (met nagatieve bijeffecten).

12.14 CGI scripts

CGI scripts

Krachtige script intepreters (shell, Perl)
Windows/NT: perl.exe
Symbolische substitutie van argumenten
& ; && ||

Notities

Een bekende klasse serverzijde WWW applicaties waarbij deze problemen zich voordoen zijn de CGI scripts. Het betreft hier serverzijde programma's in een moderne script programmeertaal (zoals PERL of shell script). Deze programma's worden op runtijd symbolisch geïnterpreteerd en uitgevoerd. De invoerparameters (uit het HTTP verzoek) worden veelal symbolisch gesubstitueerd en daarna pas bekeken. Een bekende truc is het meegeven van ";" en andere stuurtekens in parameters die door de scripttaal worden begrepen als het einde van een commando. De rest van de invoerparameter (het stuk na de ;) kan dan worden gezien als een nieuw commando. Op die manier kan een inbreker proberen zijn eigen commando's uitgevoerd te krijgen door de web server.

In Windows/NT omgevingen gebeurt het nog wel eens dat web beheerders CGI programma's in PERL schrijven. Omdat PERL echter niet standaard wordt meegeleverd met NT (en NT niet van die soepele methodes kent om scripts te koppelen aan een interpreter[1]Wat we in UNIX met #! doen), wordt "perl.exe" nog wel eens in de WWW documenten directory geplaatst. Dit is echter een behoorlijk slecht plan, omdat hackers dan door het genereren van de juiste HTTP verzoeken PERL kunnen opstarten met iedere gewenste (en vooral ook ongewenste) parameter! Op die manier is het eenvoudig mogelijk om ieder commando op een NT web server uit te laten voeren!

[1](

12.15 Oplossingen

Oplossingen

Solide coderen

Controle op invoer parameters

- Lengte
- Syntax (gebruikte karakters)

Perl taint (-T) optie

CGI/API programmaís met beperkte rechten laten draaien

chroot()

Notities

De oplossing voor problemen in WWW server applicaties zitten hem vooral in het solide coderen van die applicaties. Zo is het bijvoorbeeld van groot belang dat invoerparameters uitgebreid worden gecontroleerd op lengte en syntax (geldige karakters). PERL kent hiervoor als hulpje de “-T” optie (of de speciale “taintperl” executable). In “taint” mode is iedere waarde die door de gebruiker van buitenaf is ingevoerd automatisch verdacht. Variabelen waarvan de waarde is vastgesteld met een verdachte (tainted) variabele zijn zelf ook verdacht. Met verdachte waarden kunnen geen bestanden worden geopend of systeemaanroepen worden gedaan. Een verdachte waarde dient eerst via een speciale “untaint” operatie te worden gevalideerd.

Daarnaast is het natuurlijk altijd een goed idee om de WWW server programmatuur met minder rechten te laten draaien (niet met het root userid), of in een omgeving waarin de root directory is verlegd (zie chroot(2)).

Module 13

Secure Socket Layer

13.1 Secure Socket Layer

Secure Socket Layer

Presentatielaag gebaseerd op BSD sockets

Bedacht door Netscape

Features:

- Uitwisseling X.509 certificaten (identiteit)
- Encryptie datastroom

Client en server moeten speciale SSL software bevatten.

Notities

Vrijwel alle Internet applicaties sturen hun data klip en klaar door het netwerk. Zo ook de World Wide Web browsers en servers. Dit maakt dergelijke applicaties natuurlijk in principe ongeschikt voor het realiseren van commerciële client server applicaties. Verder vindt er in de meeste Internet applicaties geen echte wederzijdse authenticatie plaats. Als er al sprake is van enige authenticatie dan is het de client die zich bij de server moet melden met bijvoorbeeld een userid of een wachtwoord.

Deze inherente onveiligheid vormde een groot obstakel voor het gebruik van WWW-technologie voor het bouwen van transactiegeoriënteerde Internet applicaties. Alhoewel de basis daartoe met HTTP en HTML wel degelijk aanwezig was. Om deze reden ontwikkelde Netscape de Secure Socket Layer, een API en protocol voor het opzetten en gebruiken van beveiligde netwerkverbindingen. SSL ondersteunt:

1. Uitwisseling van certificaten bij het opzetten van de verbinding. Hierdoor “weten” de partners in de communicatie met wie ze communiceren.
2. Versleutelen van alle data die door de verbinding wordt geduwd.
3. “Watermerken” van de data zodat duidelijk is dat die data niet tussentijds is gewijzigd.

Om van SSL gebruik te kunnen maken moeten zowel de client als de server gebruik maken van speciale SSL libraries. In het geval van “beveiligde” WWW applicaties moeten zowel de browser als de HTTP server worden geïnstrueerd een SSL verbinding op te zetten in plaats van een gewone socket verbinding. SSL is gebaseerd op de

Internet standaard Berkeley sockets (die door alle TCP/IP compatible systemen worden ondersteund).

13.2 Versies

Versies

SSLv2
Alleen server zijde authenticatie
SSLv3
Wederzijdse authenticatie

Netscape
SSLeay (public domain)

Notities

De originele versie van SSL gaat tegenwoordig door het leven als SSL versie 2. Deze SSLv2 ondersteunde encryptie van de verbinding en server authenticatie. Meer recentelijk heeft Netscape de specificatie van SSL versie 3 vrijgegeven. De belangrijkste toevoeging in SSLv3 is de ondersteuning van client authenticatie (waarbij de client een certificaat moet tonen om de verbinding te kunnen opzetten). Client authenticatie is natuurlijk een belangrijke toevoeging voor het kunnen ontwikkelen van complete client server applicaties.

De specificatie van SSL is door Netscape vrijgegeven, en kan worden opgehaald op de Netscape World Wide Web site. Verder heeft Netscape een referentie implementatie van SSL ontwikkeld die door derden in licentie kan worden genomen. In verband met juridische problemen (ITAR restricties, zie verderop in deze module) mag deze referentie implementatie echter niet worden geëxporteerd buiten de Verenigde Staten. De Australische programmeur Eric Young heeft echter op basis van de vrij verkrijgbare specificaties een public domain versie van SSL ontwikkeld met de naam SSLeay. Deze gratis SSL versie ondersteunt zowel SSLv2 als SSLv3, en draait op alle voor de hand liggende besturingssystemen (ondermeer UNIX, Windows95 en Windows/NT).

13.3 Toepassingen

Toepassingen

World Wide Web
- https protocol (TCP poort 443)
- ìSecure serversî

Secure versies van:
- telnet
- SMTP

SRAM

Notities

De belangrijkste initiële toepassing van SSL is natuurlijk in het World Wide Web. Door het opgeven van de protocol identificatie “https” [1] in een URL kan de browser worden meegedeeld dat er een SSL verbinding met de web server tot stand moet worden gebracht (hiervoor is TCP poort 443 gereserveerd). Door die SSL verbinding wordt dan het gewone HTTP protocol gesproken. Zoals gezegd moeten dan wel zowel de client als de server SSL begrijpen.

SSL is echter veel breder van toepassing. Inmiddels zijn er ook al SSL versies van andere applicaties ontwikkeld. Op die manier genieten ook die applicaties van de voordelen van SSL: encryptie en authenticatie. Ook eigen applicaties kunnen van SSL gebruik maken. De Open Solution Providers ontwikkelt momenteel een system monitor tool (SRAM) wat gebruik maakt van SSL voor client/server communicatie.

[1] Bijvoorbeeld: https://www-secure.cdrom.com

13.4 Voorbeeld “secure” pagina

Notities

Op de slide ziet u een voorbeeld van hoe met de Microsoft Internet Explorer een web pagina is opgehaald van een secure server. Op het eerste gezicht ziet u niets bijzonders, echter bij nadere analyse vallen er twee dingen op:

1. De URL begint met “https://”.
2. Rechts onderin ziet u een gesloten slotje, wat aangeeft dat u een beveiligde pagina bekijkt.

Sommige browsers geven een waarschuwing als u van beveiligde naar niet beveiligde web servers overschakelt.

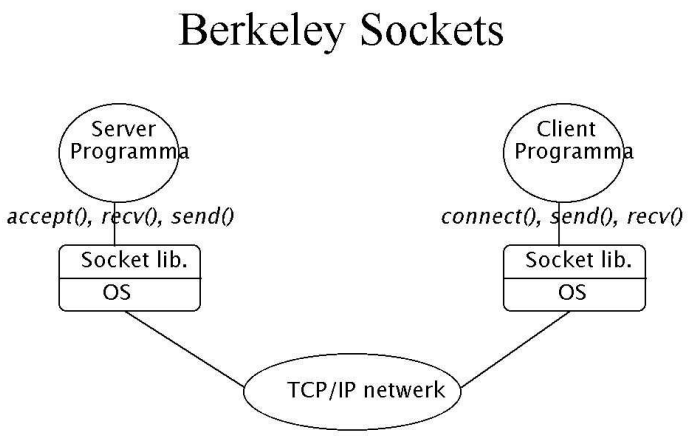
13.5 Pagina eigenschappen


en

Notities

Uw browser kan u meestal wat eigenschappen laten zien van de pagina’s die u bekijkt. In het geval van een beveiligde pagina krijgt u ook de “geloofsbrieven” te zien van de server die u de pagina heeft opgestuurd.

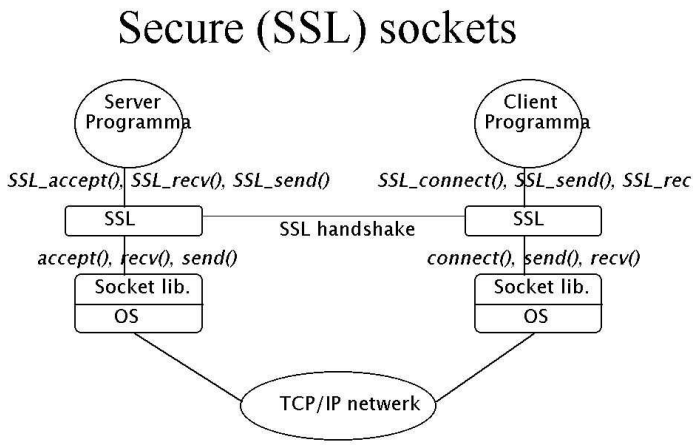
13.6 Berkeley Sockets



Notities

Normaal gesproken maken de programmeurs van Internet applicaties gebruik van de Berkeley Socket API om netwerkverbindingen te openen. Deze API bevat aanroepen zoals *bind()*, *connect()*, *send()* en *recv()* voor het manipuleren van de sockets. Berkeley sockets vertalen vrij rechtstreeks naar de onderliggende TCP/IP protocollen. De socket libraries voegen vrijwel geen functionaliteit op het gebied van de codering en presentatie van de data toe.

13.7 Secure (SSL) sockets



Notities

De Secure Socket Layer vormt een laag bovenop de Berkeley sockets. Iedere SSL socket is gekoppeld aan één Berkeley socket. De aanroepen die SSL aan de programmeur ter beschikking stelt zijn dan ook rechtstreeks afgeleid van de mogelijkheden van de Berkeley sockets. De tussenliggende SSL laag draagt zorg voor de encryptie en authenticatie, en geeft de data dan over aan de Berkeley socket laag. SSL kan dan ook worden beschouwd als OSI laag 6 (presentatielaag) functionaliteit.

Om wat meer inzicht te kunnen geven in de functies van SSL gaan we eerst in op encryptie en digitale handtekeningen.

13.8 Encryptie

Korte intro encryptie

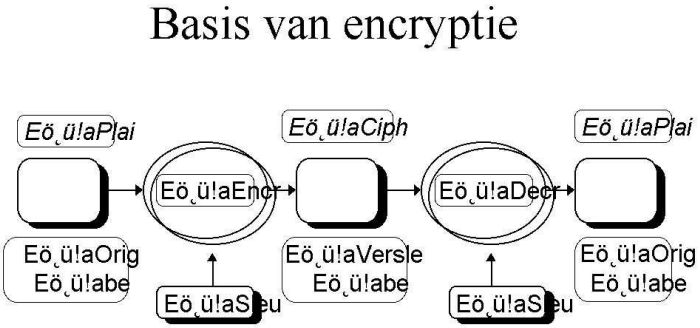
Geheimschrift
Julius Caesar
Ontwikkelingen:
MS Crypto API
HP ECF

Notities

Om de exclusiviteit en integriteit van berichten die over Internet worden verstuurd te waarborgen kunnen we gebruik maken van encryptie (versleuteling). Steeds meer besturingssystemen en Internet applicaties ondersteunen encryptie. Rondom encryptie is een hele wetenschap, cryptographie, ontstaan die zich bezighoudt met het bedenken van nieuwe encryptiealgoritmes en het breken van bestaande algoritmes.

Het doel van encryptie is het zodanig veranderen van een bericht dat er bijzondere maatregelen nodig zijn om het weer leesbaar te maken. Idealiter hebben alleen de geadresseerden van het bericht de mogelijkheid om het bericht weer leesbaar te maken. Door een bericht te versleutelen wordt de inhoud ontoegankelijk voor afluisteraars die opzettelijk of toevallig het bericht onder ogen krijgen. Encryptie is al zo oud als de mensheid zelf. Naar verluidt paste Julius Caesar encryptie toe om de communicatie tussen het front en Rome te beveiligen. Er bestaan momenteel een groot aantal encryptiealgoritmes die met meer of minder succes kunnen worden ingezet om berichten te (de)coderen.

13.9 Basis van encryptie



Notities

In het plaatje op de slide wordt het proces van encryptie schematisch weergegeven. Het originele bericht (de plaintext) wordt door het encryptieprogramma versleuteld en omgevormd tot het versleutelde bericht (de ciphertext). Deze ciphertext kan vervolgens door Internet heen worden gestuurd naar de ontvanger. Deze voert het versleutelde bericht aan het decryptieprogramma die het originele bericht weer boven water weet te toveren. Idealiter mag de ciphertext door iedereen worden onderschept en bekeken. Alleen de legitieme ontvanger heeft de beschikking over de mogelijkheden om het bericht terug te sleutelen.

Het is natuurlijk verreweg het handigst indien de encryptie- en decryptiefaciliteit zijn geïntegreerd in de applicatie die de berichten genereert en verwerkt (bijvoorbeeld het email programma of de WWW-browser). Er zijn echter ook losse encryptieprogramma's in omloop waarmee berichten buiten de applicatie om kunnen worden ont- en versleuteld. Het bekendste (en beste) publiek verkrijgbare encryptieprogramma op dit moment is PGP (Pretty Good Privacy). Meestal wordt voor de encryptie en de decryptie hetzelfde programma gebruikt (met eventueel andere opties).

De encryptie en decryptie van berichten wordt beïnvloed door twee sleutels die samen met het bericht in het encryptieprogramma moeten worden gevoerd. Het encryptieprogramma voert een wiskundige bewerking uit op het bericht en de sleutel. Bij een goed encryptieprogramma is het gebruikte algoritme bekend. Hierdoor hangt de beveiliging van het bericht af van de kwaliteit van het algoritme en de sleutel. Van een geheim algoritme kan de kwaliteit niet worden onderzocht en een dergelijk algoritme kan dus gaten bevatten die het breken van de code eenvoudiger maken.

Het doel van de inbreker is om de versleuteling te breken. Hierbij wordt gepoogd om

de ciphertext weer leesbaar te maken zonder dat men op voorhand de beschikking heeft over de juiste sleutel. Een algoritme wat relatief eenvoudig valt te breken noemen we een zwak algoritme. Sterke algoritmes zijn daarentegen moeilijk te breken of zelfs onbreekbaar (naar de huidige stand van de techniek).

13.10 Juridische aspecten van encryptie

Juridische aspecten van encryptie

Verboden in Frankrijk (sinds 1934!)
Voorontwerp van wet in Nederland (1993)
Exportverbod in U.S.A. (ITAR)
Key Recovery
Key Escrow

Notities

Alvorens dieper in te gaan op de toepassing van encryptie op Internet is het zinnig eerst enkele woorden te wijden aan de juridische aspecten van encryptie. De hedendaagse encryptiealgoritmes en computers maken het mogelijk een bericht dermate sterk te versleutelen dat het vrijwel onmogelijk is om het bericht, zonder de beschikking over de sleutel, weer leesbaar te maken.

Er zijn echter voor de overheid legitieme redenen om een versleuteld bericht weer leesbaar te willen maken. Gecodeerde telefoongesprekken tussen criminelen en versleutelde boekhoudingen van criminele aktiviteiten zijn hier voorbeelden van. Om deze redenen gaan er met de regelmaat van de klok stemmen op om encryptie aan banden te leggen. Een vorm van regulering die recentelijk onder de aandacht is gekomen is Key Escrow. Hierbij zijn partijen verplicht om de sleutels die ze gebruiken te deponeren bij een centraal depot. De overheid zou dan met toestemming van de rechter een sleutel uit het depot kunnen halen om een bericht terug te sleutelen.

De Amerikaanse overheid heeft naar analogie hiervan een encryptie chip ontwikkeld, de Clipper chip, waarbij de sleutels die nodig zijn voor de decryptie automatisch (tijdens de fabricage van de chip) bij een centrale instantie worden belegd. De Clipper chip wordt vrijelijk (tegen lage kosten) beschikbaar gemaakt voor industriële en bedrijfstoepassingen.

Naar mijn inzicht zal deze methode onwerkbaar zijn en niet het gestelde effect bereiken. Criminelen zijn al crimineel en het is mijns inziens onzinnig om te veronderstellen dat een crimineel zijn sleutels bij het centrale depot zal beleggen. Vervolgens hoef je in Nederland niet aan je eigen veroordeling mee te werken. Het vooraf verplicht deponeren van decryptiesleutels valt wellicht onder het verplicht meewerken aan een

mogelijke veroordeling. Ook zijn er encryptiealgoritmes bekend die als ciphertext een gedicht opleveren. Van de buitenkant is dan niet te zien dat het hier een versleuteld bericht betreft!

In de Verenigde Staten valt encryptieprogrammatuur onder de wapenwet (de ITAR, in de categorie munitie). Hierdoor moeten leveranciers die applicaties willen exporteren waarin encryptiemogelijkheden zijn opgenomen een exportvergunning aanvragen. De geldende praktijk is dat alleen zwakke algoritmes toestemming krijgen voor export. Hierdoor kunnen Amerikaanse software producenten als Microsoft, Lotus en Netscape geen sterke encryptiemogelijkheden in hun software bieden. Een Franse onderzoeker toonde dit korte tijd geleden aan door binnen enkele weken een bericht wat met Netscape was versleuteld te breken. De houding van de Amerikaanse overheid in deze bevordert de beveiliging van Internet dataverkeer niet!

13.11 Het breken van encryptie

Het breken van encryptie

Cryptanalyse
Sleutels stelen
Zwakheden in de encryptie algoritmes
Brute Force

Notities

De veiligheid die door encryptie wordt geboden is natuurlijk geheel afhankelijk van de moeite die moet worden gedaan om het gekozen algoritme te breken. Bij het breken van een versleuteld bericht proberen we de inhoud van het bericht te achterhalen zonder dat we op voorhand de sleutels kennen.

13.11.1 Sleutels stelen

Verreweg de meest effectieve methode om een bericht te ontsleutelen is door het stelen van de sleutel die nodig is om het bericht leesbaar te maken. In veel gevallen bewaren gebruikers sleutels in bestanden op hun computer. Een inbreker die zich op andere wijze toegang heeft verschaft kan de sleutels uit het bestand kopiren zonder dat de gebruiker weet dat zijn sleutels niet langer geheim zijn. Het PGP programma versleutelt om die reden het bestand met sleutels met een extra sleutel (de pass phrase). Ook het onderscheppen van sleutels tijdens transport (bijvoorbeeld per email) behoort tot de mogelijkheden.

13.11.2 Zwakheden in het algoritme

Sommige algoritmes bevatten inherente zwakheden die het mogelijk maken een bericht geheel of gedeeltelijk leesbaar te maken. Een eenvoudig voorbeeld van zo'n zwakheid is een algoritme die de statistische samenhang van een bericht bewaart. Van de Nederlandse taal is per letter bekend hoe vaak die gemiddeld in een bericht voorkomt. Ook van combinaties van letters is frequentie waarmee die voorkomen uitgezocht. Oudere

encryptiealgoritmes (op het niveau: Donald Duck) bewaarden deze statische samenhang in de ciphertext. Hierdoor kon met een statistische analyse van de ciphertext de originele inhoud van het bericht worden geraden. Moderne cryptanalysis technieken zijn natuurlijk vele malen geavanceerder maar proberen ook dergelijke zwakheden in een algoritme uit te buiten.

13.11.3 Brute force

Een zekere manier om binnen een vastgestelde tijd de originele inhoud van een versleuteld bericht te achterhalen is het domweg proberen van alle mogelijke sleutels. Een nadeel van een dergelijke brute kracht aanval is dat het afhankelijk van de lengte van de sleutel is hoe lang het duurt voordat het bericht in zijn originele staat is teruggebracht. Voor een algoritme waar nog geen zwakheden van bekend zijn is brute force echter de enige mogelijke aanvalstechniek. Door een sleutel van voldoende lengte te kiezen kunnen we de tijd die nodig is voor een dergelijke aanval tot ongekende hoogte opvoeren.

Met een sleutel van 16 karakters (128 bits) en een computer die honderd miljoen sleutels per seconde kan proberen zijn we gemiddeld 53.951.415.354.000.000.000.000 jaar bezig voor we het originele bericht hebben achterhaald. Dezelfde computer doet er echter maar 33 dagen over om alle mogelijke sleutels van 6 karakters (48 bits) te proberen!

De Amerikaanse overheid geeft naar verluidt geen exportvergunningen af voor encryptiealgoritmes die met sleutels werken die langer zijn dan 40 bits.

13.12 Bekende encryptiealgoritmes

Bekende encryptiealgoritmes

Symmetrische algoritmes:
DES
Triple-DES
RC2 en RC4
IDEA

Notities

In de loop der jaren zijn er een grote hoeveelheid encryptiealgoritmes verzonnen en weer afgezonken (in verband met zwakheden). Wij zijn voorstander van het gebruiken van een algemeen bekend encryptiealgoritme waarvan de kwaliteit vast staat en aan wetenschappelijk onderzoek onderhevig is.

De meeste encryptiealgoritmes zijn symmetric private key algoritmes. Dit betekent dat voor de encryptie en decryptie dezelfde geheime sleutel wordt gebruikt. Het grote probleem met deze klasse van algoritmes is het distribueren van de sleutels. Alvorens zo'n algoritme kan worden gebruikt dienen zender en ontvanger beiden dezelfde geheime sleutel te kennen. Deze sleutel moet dus op een veilige wijze kunnen worden uitgewisseld. Door het inzetten van een leger van koeriers die ieder met een gedeelte van de sleutels op pad gaan kan dit probleem worden opgelost.

13.12.1 DES

Het DES (Data Encryption Standard) algoritme vormde jarenlang de encryptiestandaard van de Amerikaanse overheid en het internationale zakenleven. Het algoritme gebruikt een sleutel van 56 bits (7 karakters) en is dientengevolge vandaag aan de dag gevoelig voor brute force aanvallen. Michael Wiener van Bell Northern Research toonde aan dat voor tien miljoen dollar een computer kan worden gebouwd die ieder bericht wat met DES is versleuteld in 21 minuten weer leesbaar kan maken.

13.12.2 Triple-DES

In een poging om het leven van DES nog ietwat te rekken is Triple-DES ontwikkeld. Hierbij wordt het DES algoritme drie keer uitgevoerd met twee verschillende sleutels. Het resulterende algoritme kan worden bezien als een versie van DES met een effectieve sleutel van 112 bits.

13.12.3 RC2 en RC4

Deze encryptie algoritmes (uitgevonden door professor Ronald Rivest) worden momenteel tamelijk algemeen toegepast in Internet. De algoritmes werken met een variabele sleutellengte van 1 tot 1024(!) bits. Indien sleutels korter dan 48 bits (6 karakters) worden gebruikt zijn de algoritmes relatief eenvoudig te breken. Doordat de algoritmes intellectueel eigendom zijn van RSA Data Security Inc. weet niemand precies hoe veilig ze zijn bij langere sleutels. Netscape gebruikt (onder andere) RC4 met een sleutellengte van 40 bits als SSL encryptieprotocol in de export versie van de Netscape Navigator.

13.12.4 IDEA

Een nieuwe ster aan het encryptiefirmament is het International Data Encryption Algorithm van James Massey en Xuejia Lai. In tegenstelling tot de hierboven ontwikkelde algoritmes betreft het hier een Europees (Zwitsers) produkt. Het encryptieprogramma PGP gebruikt onder andere IDEA voor het versleutelen van berichten. Voor zover nu bekend bevat IDEA geen zwakheden. Het algoritme werkt met sleutels van 128 bits (16 karakters).

13.13 Public Key Encryption

Public Key Encryption

Asymmetrische algoritmes
Bijvoorbeeld: RSA
Publieke sleutel:
- Nodig voor encryptie
- Algemeen bekend

Private sleutel
- Nodig voor decryptie
- Geheim

Notities

Een bijzondere vorm van encryptie wordt gevormd door de Public Key Encryption (PKE) methodes. Bij PKE is er sprake van een sleutelpaar bestaande uit twee verschillende sleutels. Een bericht wat met de ene sleutel wordt versleuteld kan alleen met de bijbehorende andere sleutel weer leesbaar worden gemaakt. Één van de twee sleutels, de zogenaamde public key, wordt publiekelijk bekend gemaakt. De andere sleutel, de private key, wordt ten strengste geheim gehouden.

Mensen die mij een bericht willen versturen versleutelen dit bericht met mijn public key. Eenmaal versleuteld kan het rustig over Internet worden verzonden want alleen ik kan het bericht weer leesbaar maken met mijn private key. Merk op dat zelfs de originele verzender een eenmaal versleuteld bericht niet meer kan terugsleutelen!

Het bekendste PKE algoritme is dat van Rivest, Shamir en Adleman: het RSA-algoritme. RSA encryptie wordt breed toegepast door onder andere Netscape, Lotus Notes en PGP. Het bedrijf RSA Data Security Inc. heeft momenteel het alleenrecht op de verkoop en toepassing van public key encryption en het RSA-algoritme in de Verenigde Staten.

Het grote voordeel van PKE is dat er vrijwel geen sleuteldistributieproblemen zijn. De sleutel die nodig is om een bericht te versleutelen, de publieke sleutel, kan algemeen bekend worden gemaakt. Publieke sleutels kunnen vrijelijk worden gepubliceerd en gekopieerd zonder dat de inhoud van de berichten in gevaar komt.

De beveiliging van een PKE versleuteld bericht hangt volledig af van de geheimhouding van de priv sleutel. Als die eenmaal op een of andere manier bekend is geworden kan een inbreker alle berichten die voor het slachtoffer bestemd zijn weer leesbaar maken. Daar PKE sleutels tamelijk lang kunnen worden (PGP kan met sleutels van

1024 bits werken!) worden ze meestal opgeslagen in bestanden. Om te voorkomen dat die bestanden worden gestolen dienen die weer apart te worden beveiligd door ze te versleutelen met een private key algoritme als IDEA.

Een probleem wat zich met PKE kan voordoen is dat de versleutelaar van een bericht er wel zeker van moet zijn dat de gebruikte publieke sleutel daadwerkelijk bij de geadresseerde hoort. Een inbreker die het dataverkeer van een slachtoffer wil afluisteren zou kunnen proberen valse publieke sleutels van het slachtoffer in omloop te brengen. Zouden die valse sleutels worden gebruikt voor het versleutelen van berichten dan kan alleen de inbreker (die de bijbehorende privé sleutel heeft) het bericht weer leesbaar maken. Op Internet zijn er een aantal vertrouwde centrale instanties (Trusted Key Authorities) die publieke sleutels publiceren.

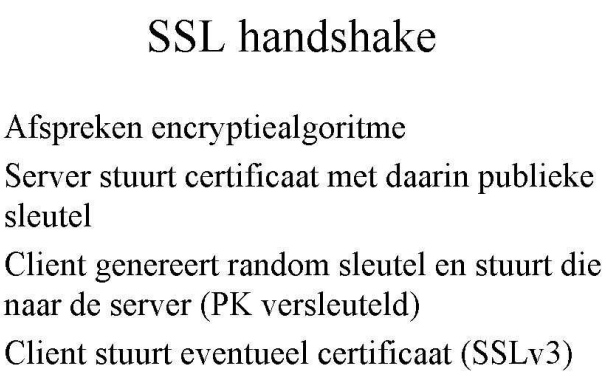
13.14 Digitale handtekeningen

Notities

Public key encryption kan ook worden gebruikt om de ontvanger van een bericht ervan te overtuigen dat de vermeende afzender ook de echte afzender is. Hiertoe dient de verzender van een bericht een digitale handtekening te plaatsen. Door een bericht te versleutelen met mijn privé sleutel geef ik dat bericht een unieke status. Iedereen ter wereld kan het bericht terugsleutelen met mijn publieke sleutel, maar het feit dat dit kan bewijst dat ik de afzender was. Alleen met mijn privé sleutel kunnen namelijk berichten worden versleuteld die met mijn publieke sleutel zijn terug te sleutelen!

Bij het veilig verzenden van een bericht vindt er dus dubbele encryptie plaats! Eerst signeer ik het bericht (versleuteling met mijn privé sleutel) en daarna versleutel ik het met de publieke sleutel van de geadresseerde. Digitale handtekeningen zijn vrijwel onmisbaar op Internet, waar immers vrijwel iedereen (potentieel) de afzender van een bericht kan vervalsen.

13.15 SSL handshake



Notities

Bij het opzetten van een SSL verbinding (met de functies uit de SSL libraries) wisselen de SSL-lagen in de client en de server eerst wat informatie uit omtrent elkanders identiteit, de te gebruiken compressie/encryptie en welke sleutels moeten worden gebruikt (de SSL handshake).

Een SSL verbinding komt als volgt tot stand:

1. De client opent de verbinding en stuurt een *client hello*. In dit bericht staat (onder andere) welke encryptie algoritmes de client begrijpt.

2. De server accepteert de verbinding en stuurt een *server hello*. Hierin staat ondermeer welk encryptiealgoritme de server met de client wil praten.

3. Onmiddelijk hierna stuurt de server een server certificate. Hiermee geeft de server aan wat zijn public key is.

4. De client kan nu een client certificate sturen (SSLv3). Deze bevat de public key van de client.

5. De client genereert een random sessie key, encrypt deze met de public key van de server (uit het server certificaat) en stuurt die op. Dit is de sleutel waarmee de sessie zal worden versleuteld (*client key exchange* en *server key exchange* berichten).

6. De client genereert vervolgens een MD5 hashcode over alle handshake meldingen, signeert die met zijn eigen private key en stuurt die naar de server. Met

dit *certificate verify* bericht kan de server controleren of de client de rechtmatige bezitter is van de public key (expliciete verificatie, alleen in SSLv3).

13.16 *Man in the Middle*

Man in the Middle

Ziet publieke key van server voorbij komen
Maar die kende hij waarschijnlijk al!
Ziet versleutelde sessie key voorbij komen
Maar kan die niet ontsleutelen omdat hij de private sleutel van de server niet kent
Rest van de sessie versleuteld met de onbekende sessiesleutel

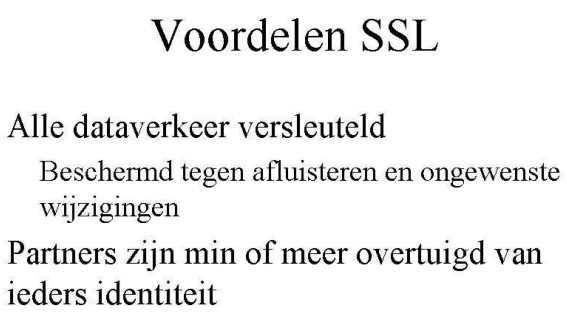
Notities

SSL verbindingen zijn beschermd tegen zogenaamde *Man in the Middle* aanvallen, waarbij een inbreker de netwerkbekabeling kan aftappen en daardoor het netwerkverkeer kan afluisteren of zelfs wijzigen. SSL verbindingen worden versleuteld met een (random bepaalde) sessiesleutel die onder public key encryptie wordt uitgewisseld. De *Man in the Middle* (MiM) ziet eerst de publieke sleutel van de server voorbij komen. Die kende hij echter waarschijnlijk als, dus dit geeft hem geen informatie. Vervolgens genereert de client een random sessiesleutel die versleuteld met de publieke sleutel van de server naar diezelfde server wordt verstuurd. De MiM kan echter die sessiesleutel niet terugsleutelen omdat daarvoor de private sleutel van de server nodig is! De server kent natuurlijk wel zijn eigen private sleutel, en hij kan dus de sessiesleutel achterhalen. Vervolgens kunnen client en server hun verdere datastroom met die geheime sessiesleutel versleutelen. De enige kans die de MiM nog heeft is het brute force terugsleutelen van de data[2].

LET OP: De exportversies van Amerikaanse software mogen alleen gebruik maken van encryptiealgoritmes met sterk gelimiteerde sleutellengtes. Zo mogen de internationale varianten van WWW browsers en servers slechts symmetrische encryptiealgoritmes gebruiken met sleutels tot een maximum van 40 bits. Het binnen afzienbare tijd brute force terugsleutelen van die data behoort binnen ieders mogelijkheden!

[2] Of gebruik maken van zwakheden in het encryptiealgoritme

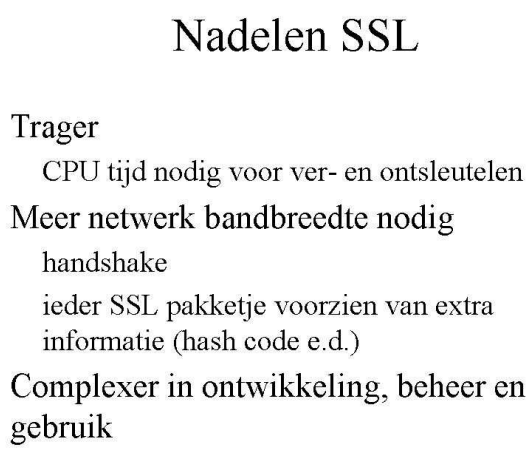
13.17 Voordelen SSL



Notities

De voordelen van het gebruik van SSL liggen voor de hand. Alle data verkeer is versleuteld en de partners kunnen (bij juist gebruik van certificaten) overtuigd zijn van elkaars identiteit. Een groot voordeel van SSL is dat we veilig kunnen communiceren met iemand die we tot nu toe niet kenden. Door het gebruik van “public key encryption” kunnen we veilig een geheime sessiesleutel afspreken!

13.18 Nadelen SSL



Notities

Het gebruik van SSL kent echter ook een aantal nadelen. Zo is er CPU tijd nodig voor het versleutelen en terugsleutelen van de data. Ook is er (iets) meer netwerkcapaciteit nodig voor het versturen van de extra SSL informatie (zoals de handshake en de hash code in ieder SSL pakket) Verder is de ontwikkeling en het beheer van applicaties die van SSL gebruik maken ingewikkelder, bijvoorbeeld door het gebruik van identiteits certificaten.

13.19 Echter, . . .

Echter, ...

Geen bescherming tegen valse server

Vertrouw ik de server?

Man in the Middle kan initiele publieke sleutel vervangen

Oplossing: X.509 digitale certificaten

Notities

In het systeem zoals dat tot nu toe is beschreven zit echter nog een zwakheid: “Waarom vertrouwen we de identiteit en de public key die de server ons toezendt?” Indien iemand een valse server heeft neergezet dan zal hij ons natuurlijk ook een valse identiteit en een valse publieke sleutel toezenden. Verder zou een hacker die “op de netwerkkabel zit” de identiteit en/of de sleutel tussentijds kunnen vervangen.

We moeten dus een manier hebben om te weten of de identiteit en de publieke sleutel die we toegezonden krijgen ook daadwerkelijk behoren bij de server met wie we denken te praten. Dit zou kunnen als we de publieke sleutel van iedere server met wie we willen praten eerst via een andere manier (bijvoorbeeld op een floppy per post) hebben verkregen. Het grote nadeel daarvan is echter dat we dan alleen kunnen praten met servers van wie we de publieke sleutel kennen!

De echte oplossing zit hem in het gebruik van gegarandeerde identificatie methodes.

13.20 X.509 Certificaat

Notities

De ISO standaard X.509 beschrijft certificaten waarmee iemands identiteit en publieke sleutel kunnen worden vastgelegd. Het bijzondere aan deze certificaten is dat ze worden afgegeven en digitaal worden gesigneerd door deze afgever (de *Issuer*). Een X.509 certificaat bevat:

1. De naam van de houder (het subject) van het certificaat.
2. De naam van de afgever (de issuer).
3. De geldigheidsperiode van het certificaat.
4. De publieke sleutel van de houder.
5. De digitale handtekening door de afgever.

De namen van de houder en afgever zijn in de vorm van X.500 *Distinguished Names* (DN's). Deze DN's bevatten diverse componenten waarin de organisaties, de plaats en het land van houder en afgever kunnen worden opgenomen. Het *Common Name* veld van de DN wordt vaak gebruikt om een server naam of een email adres te bevatten. Bij https verbindingen eist de browser bijvoorbeeld dat de common name van het subject in het certificaat dat de server presenteert overeenkomt met de volledig gekwalificeerde Internet hostnaam van de server.

De tekstuele representatie van een certificaat kan er als volgt uitzien:

```
Certificate:
```

```
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=NL, ST=NH, L=Diemen, O=Open Solution Providers,
CN=OSP SRAM root certificate/Email=sram@osp.nl
        Validity
            Not Before: Aug  6 09:16:46 1997 GMT
            Not After : Aug  6 09:16:46 1998 GMT
        Subject: C=NL, ST=NH, L=Diemen, O=Open Solution Providers,
CN=OSP SRAM test client
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d5:ca:94:98:f3:fd:f5:3a:29:5f:cf:a2:0f:ad:
                    ea:70:2b:4b:a7:d6:94:b8:c2:67:1d:40:ea:a2:52:
                    17:48:ca:12:34:90:3b:ab:05:42:aa:a4:8c:06:c1:
                    07:a1:d9:7f:a1:3d:ff:60:fe:77:7a:eb:67:6a:9d:
                    12:81:ca:3c:4a:27:0b:e3:da:ee:b7:c0:70:4b:cd:
                    2c:e5:62:a5:bd:ee:56:39:67:fa:49:8a:58:d2:85:
                    04:93:64:0b:fb:78:bf:73:ea:66:64:f6:36:9f:54:
                    89:d9:6c:da:5b:24:9d:c5:e8:07:e7:b5:3e:5c:8f:
                    67:d3:0a:d8:c0:67:04:bb:13
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape CA Revocation Url:
                http://www.osp.nl/ca-crl.pem
            Netscape Comment:
                No comment
            Netscape Cert Type:
                0x40
    Signature Algorithm: md5WithRSAEncryption
        0d:dc:08:41:7f:83:f8:bc:a7:fb:06:be:a8:c1:c7:ab:b3:10:
        12:58:4e:b5:07:a7:86:a8:14:b6:91:ef:df:f7:06:9d:b1:9b:
        8e:ec:6e:a6:1f:84:e3:f3:6c:21:c4:c8:39:40:75:df:13:3c:
        c4:8c:25:a0:f4:d4:52:9f:aa:a8:18:8a:10:0b:ad:99:3f:49:
        2e:69:38:1a:99:a1:37:19:a3:75:32:05:88:59:4a:85:6e:ca:
        f2:60:c9:e0:24:ad:53:60:00:d1:e6:d8:a2:5b:6f:d9:e0:03:
        4e:61:27:fc:85:a9:68:81:06:70:c9:87:23:20:c6:e3:1d:84:
        40:a4
```

SSLeay (de public domain implementatie van SSL door Eric Andrew Young) bevat tools voor het genereren en signeren van certificaten. Netscape verkoopt de “Netscape Certificate Server” waarmee hetzelfde kan.

13.21 Wanneer vertrouw ik een certificaat

Wanneer vertrouw ik een certificaat?

Als ik het certificaat al ken.

Als ik het certificaat van de *issuer* ken.

Als het certificaat van de *issuer* is getekend door iemand die ik ken.

Recursieve verificatie!

Speciale *issuers:*

Certifying Authority

Trusted Third Parties

Notities

Zoals gezegd is een X.509 certificaat digitaal gesigneerd door de afgever. Dit betekent dat de afgever een hash code over het certificaat heeft berekend en deze met zijn privésleutel heeft versleuteld. De ontvanger van het certificaat kan de digitale handtekening controleren door de handtekening te decrypten en deze te vergelijken met de hash code van het certificaat (opnieuw berekenen). Hiervoor dient de ontvanger echter wel de publieke sleutel van de afgever te kennen!

Als ik de publieke sleutel van de afgever van een certificaat niet ken, kan ik de digitale handtekening op het certificaat niet controleren. De server kan mij echter ook een X.509 certificaat van de afgever doen toekomen. Hierin zit de publieke sleutel van de afgever, waarmee ik dan het originele certificaat kan controleren. Echter, dat digitale certificaat van de afgeven vertrouw ik natuurlijk alleen als dat certificaat weer is afgegeven door iemand (de meta-afgever) wiens publieke sleutel ik al ken. Mocht dat niet het geval zijn, dan kan de server natuurlijk het X.509 certificaat van de meta-afgever opsturen. Enzovoorts...

Het controleren van een certificaat kan dus leiden tot een recursieve verificatie van de certificaten van de afgevers. Deze recursieve verificatie stopt zodra ik een certificaat tegenkomen wat is afgegeven door iemand die ik al ken. Het signeren van een certificaat is dus een teken van *vertrouwen*. Als ik A vertrouw (doordat ik zijn certificaat op een of andere manier ben bekomen), en B heeft een certificaat heeft dat getekend is door A, dan vertrouw ik dat certificaat van B, alhoewel ik B nog nooit eerder ben tegengekomen!

Als ik dus contact leg met een server, en die server presenteert een certificaat, dan ik het dus van groot belang dat in de certificatenketting van die server iemand zit wiens

certificaat ik al ken.

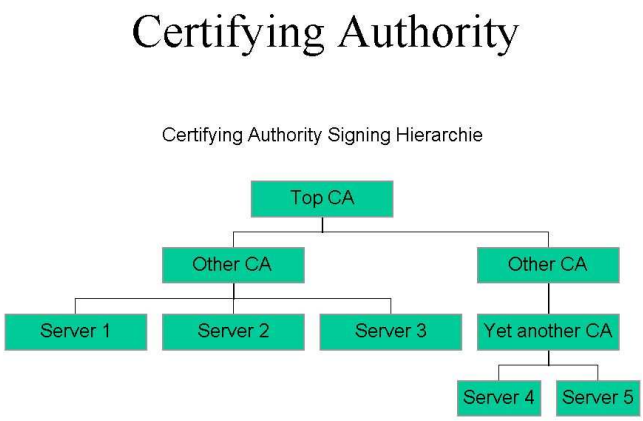
Er zijn al diverse instanties opgericht die zich er speciaal op richten om andermans certificaten te tekenen. We noemen dergelijke instanties *Certifying Authorities* (CA's) of *Trusted Third Parties*. Dergelijke instanties moeten natuurlijk het vertrouwen genieten van partijen in de markt. Als u een WWW browser installeert dan krijgt u met die browser gelijk een aantal certificaten mee van bekende CA's.

13.22 Wie vertrouwt u?

Notities

Op de slide ziet u een voorbeeld van de “Certificate Store” van een WWW browser. Deze certificaten zijn reeds door u geaccepteerd (allemaal met de browser mee geïnstalleerd. Nieuwe certificaten kunnen worden geïnstalleerd van een floppy of via het WWW worden opgehaald (liefst van een veilige server).

13.23 Certifying Authority



Notities

Door de recursie die inherent is aan het controleren van de certificaten ontstaan er hierarchieën van CA's die elkaars certificaten hebben gesigneerd. Aan de top van de hierarchie zit een "root certificate". Zo'n certificaat is door zichzelf gesigneerd[3]. Een dergelijk certificaat kan nooit worden gecontroleerd maar moet altijd rechtstreeks worden geaccepteerd.

Het grote voordeel van deze hierarchie is dat iedereen alleen zijn eigen certificaat het het certificaat van de "Top CA" hoeft te kennen om toch alle certificaten in de hierarchie te kunnen valideren!

[3] ook wel een "self signed certificate" genoemd

13.24 Niet verifieerbaar certificaat?

Niet verifieerbaar certificaat?

Browser afhankelijk
- Foutmelding
- Keuze wel/niet doorgaan

Notities

Wat er gebeurt als SSL een certificaat ontvangt wat niet kan worden geverifieerd is afhankelijk van de SSL implementatie en de applicatie. De mogelijkheden zijn:

1. Verbinding verbreken
2. Verbinding verbreken afhankelijk van interactie met de gebruiker
3. Verbinding aannemen met een waarchuwing

Het beveiligingsoverwegingen is de eerste keuze natuurlijk de beste. Een probleem met de tweede mogelijkheid is dat het een belangrijke beveiligingskeuze in de handen van de gebruiker legt.

13.25 Gebruik van SSL

Notities

Om gebruik te kunnen maken van SSL dient men allereerst over een web server te beschikken die SSL compatibel is. Vrijwel alle commerciële web servers kunnen dit trouwens. Bij die web server zitten vervolgens hulpmiddelen waarmee een public/private sleutelpaar en een X.509 “Certificate Request” kan worden aangemaakt. Zo’n request is een half ingevuld certificaat wat vervolgens naar de CA moet worden gezonden. Deze tekent het certificaat (na het overmaken van de benodigde blijheidspenninkjes) en stuurt het weer terug. Ook is het mogelijk om het certificaat zelf te ondertekenen. Hiertoe dient u een eigen CA te hebben ingericht met bijvoorbeeld de Netscape Certificate Server.

Het aldus verkregen certificaat dient weer in de web server te worden geïnstalleerd. Vanaf dat moment kunnen klanten via SSL aan de web server aankoppelen. Het certificaat van de CA dient vervolgens wel rechtstreeks of indirekt aan de klanten bekend te worden gemaakt!

Deel IV

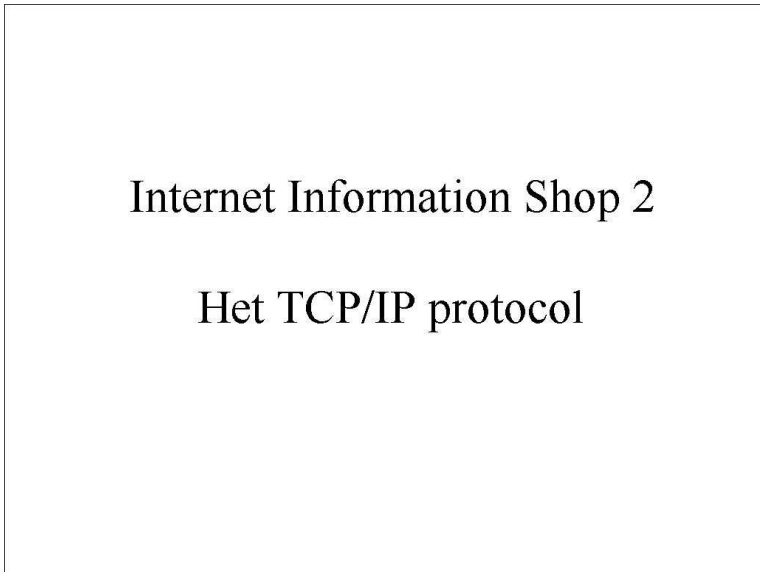
Appendices

Bijlage A

Introduktie TCP/IP

Deze appendix bevat een introduktie in TCP/IP, en legt diverse TCP/IP onderwerpen zoals IP-adres, subnet masker en domain namen verder uit. Dit cursusmateriaal is overgenomen uit de OSP “Internet Information Shops”, een vijfdelige reeks mini-cursussen (van één middag per stuk) over uiteenlopende Internet onderwerpen.

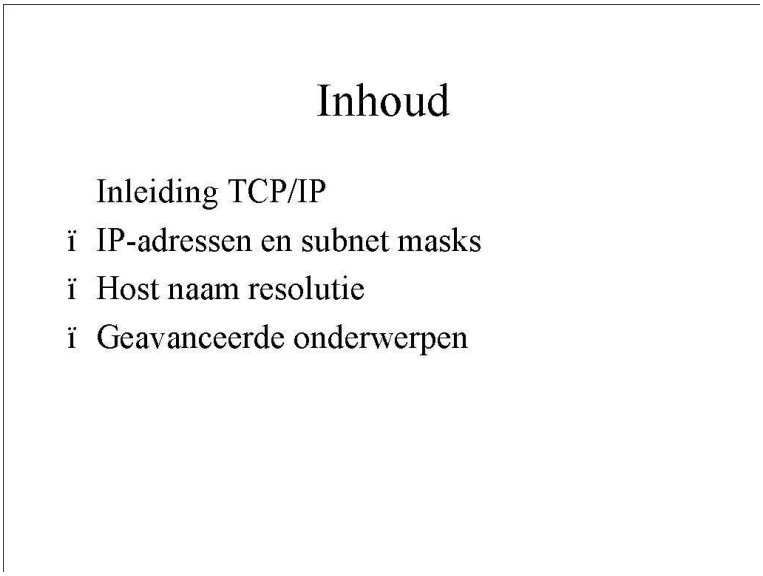
A.1 Welkom



Notities

Welkom bij deze tweede Internet Info Shop. In deze info shop gaan we dieper in op het TCP/IP protocol. Deze shop is niet bedoeld voor systeembeheerders die uitgebreid willen weten hoe TCP/IP op hun specifieke computer dient te worden geconfigureerd. De nadruk ligt op het beschrijven van de concepten en implementatie van TCP/IP. Enige technische diepgang wordt daarbij niet geschuwd, maar staat niet centraal.

A.2 Inhoud



Notities

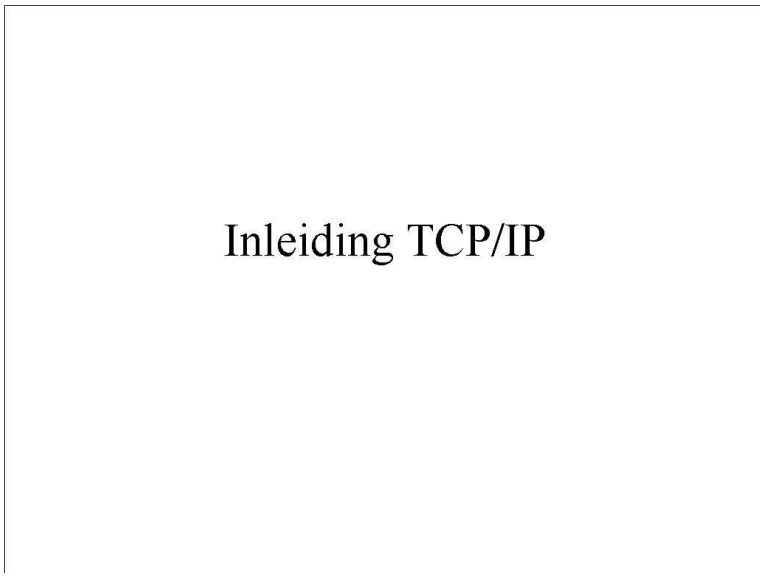
In de shop gaan we in op:

- De structuur van TCP/IP
- De adressering en naamgeving van nodes in TCP/IP netwerken
- Hoe hostnamen worden gekoppeld aan IP-adressen en omgekeerd
- TCP en UDP poortnummers
- Routering in TCP/IP netwerken
- Sockets
- IP next generation

De laatste vier onderwerpen hebben het label “geavanceerd” gekregen en zullen alleen worden behandeld indien het verloop van de shop daar gelegenheid toe laat.

A.3 Inleiding TCP/IP

A.4 Inleiding



Notities

In deze module wordt ingegaan op de noodzaak en structuur van netwerkprotocollen in het algemeen, en TCP/IP in het bijzonder.

A.5 Het probleem

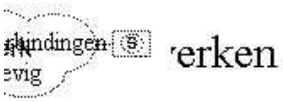
Notities

De kern van de gehele netwerkproblematiek kan worden omschreven als de wens van twee applicaties die niet in dezelfde computer draaien om informatie uit te wisselen. Denk hierbij aan:

- Het raadplegen van een remote database
- Het versturen van een electronisch postbericht
- Het versturen van een document naar een remote printer
- Het kopiëren van bestanden van een computer naar een andere

Daar de applicaties niet in dezelfde computer draaien zijn de traditionele wijzen van gegevensuitwisseling tussen applicaties (via diskbestanden of het interne geheugen) niet van toepassing. Indien beide computers fysiek met elkaar zijn verbonden zal de uitwisseling plaats moeten vinden over het netwerk. Via speciale hardware kunnen bytes worden omgezet in electrische of optiesche signalen en vice versa. Dit biedt in principe de mogelijkheid om twee applicaties gegevens uit te laten wisselen.

A.6 Eigenschappen van netwerken



Notities

Netwerken hebben echter een aantal zeer vervelende eigenschappen:

1. Complexe structuur
   In een groot aantal gevallen zullen de computers waarin de applicaties draaien niet rechtstreeks met elkaar verbonden zijn. Netwerken hebben meestal een complexe structuur en om succesvol te kunnen communiceren dient de informatie dus door het netwerk te worden geloodst. Het zoeken van de optimale route door het netwerk is niet triviaal.

2. Storingsgevoelig
   Netwerken bestaan uit allerlei soorten verbindingen op electrische, electromagnetische en optische basis. Deze verbindingen zijn gevoelig voor magnetische en fysieke verstoringen. De data die dan toevallig over de lijn reist kan dan geheel of gedeeltelijk worden verminkt. Ook het tijdelijk of permanent uitvallen van de verbinding behoort tot de mogelijkheid.

3. Verschillende soorten verbindingen
   Netwerken zijn vaak opgebouwd uit zeer verschillende soorten verbindingen. Denk hierbij an telefoonlijnen, ISDN, ethernet segmenten, token ringen, microgolf radio verbindingen en glasvezel kabels. Al die verschillende “kabels” moeten op hun eigen specifieke wijze worden aangestuurd en kennen hun eigen mogelijkheden, snelheden en beperkingen.

4. Aan verandering onderhevig
   De structuur van een groot netwerk verandert continue. Er komen verbindingen bij, de verkeersdrukte varieert, verbindingen vallen weg of krijgen juist een grotere capaciteit.

De wijze waarop informatie door het netwerk moet worden gestuurd moet dus met al deze factoren rekening houden. Vanzelfsprekend wenst een applicatieprogrammeur of gebruiker niet met deze achterliggende technische zaken te worden vermoeid!

A.7 De oplossing: netwerkprotocollen

De oplossing: netwerkprotocollen

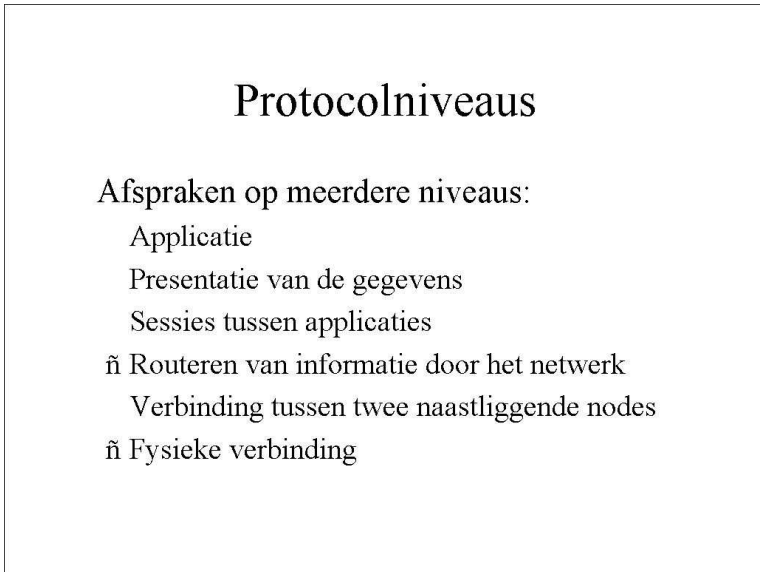
- Afspraken over:
  - Vorm en inhoud berichtenuitwisseling
  - Identificaties nodes en applicaties (netwerkadressering)
  - Verzending van berichten door het netwerk (routering, prioriteit)
  - Ontvangstbevestiging en herverzending
  - Fysieke toegang tot het netwerk
  - Maximale grootte van de pakketten

Notities

De oplossing voor al deze problemen wordt gevormd door het ontwikkelen van specifieke stukken programmatuur die de uitwisseling van gegevens via het netwerk voor hun rekening nemen. Deze netwerkprogrammatuur dient alle met de verzending samenhangende problemen te ondervangen. Deze programmatuur moet natuurlijk onderling goed samenwerken. Het is van groot belang dat de netwerkprogrammatuur van de communucatiepartners hetzelfde idee hebben over hoe de communicatie is gestructureerd.

Deze afspraken noemen we *netwerkprotocollen*. In de netwerkprotcollen wordt uitgebreid vastgelegd hoe de verzending van informatie door het netwerk plaatsvindt.

A.8 Protocolniveaus



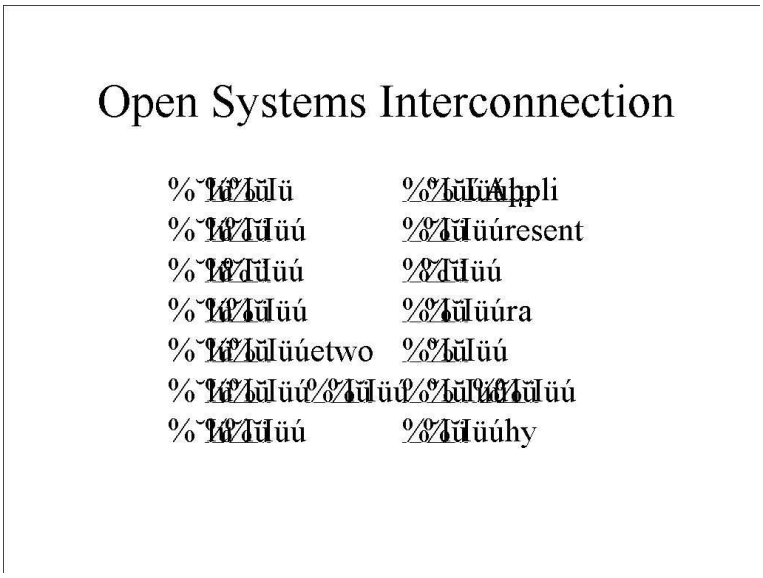
Notities

In het verleden is er wel netwerkprogrammatuur ontwikkeld volgens het “kluitmodel”: alle functies van de communicatie geïntegreerd in één groot programma. Al snel kwam men erachter dat dit niet erg handig is (bijvoorbeeld voor onderhoud en interoperabiliteit). De problematiek van het verzenden van gegevens over netwerken kan op verschillende abastractieniveau’s worden bekeken en opgelost:

1. Het via een kabel verzenden van bits
   Afspraken over el(ectrische verschijnselen, timings, weerstand.
2. Het verzenden van een pakketje informatie naar een naastliggende node (één hop verwijderd))
3. Het doorverzenden van pakketjes die niet voor deze node zijn bestemd
   (Routeren van pakketjes)
4. Het verzenden van een informatiestroom tussen twee applicaties
   (O.a. ervoor zorgen dat de pakketjes in de juiste volgorde aankomen)
5. Het maken van afspraken over de presentatie van de gegevens
   (Karaktercodering (ASCII, EBCDIC), encryptie, compressie).

Mits goed opgezet ontstaat op deze wijze een bouwwerk van oplossingen die gebaseerd zijn op de onderliggende verdiepingen. We noemen een dergelijk bouwwerk ook wel een *protocolstapel*.

A.9 Open System Interconnection



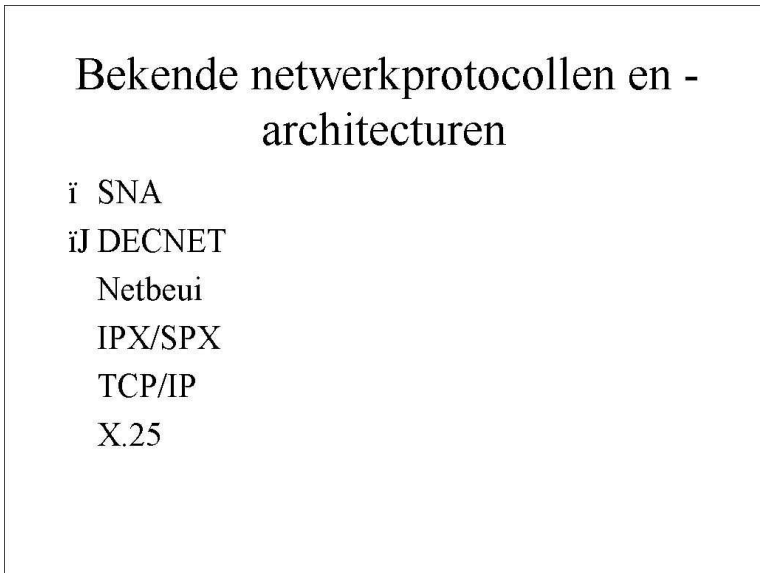
Notities

Gebaseerd op de ideeën van protocolstapels is er door de internationale standaardiseringsorganisatie ISO een raamwerk afgesproken voor het beoordelen van netwerkprotocollen. Dit raamwerk staat alom bekend als het OSI (Open Systems Interconnection) model. Dit OSI model is een formele weerslag van de niveaus waarop netwerkcommunicatie tussen twee applicaties betrekking heeft.

Voor het onthouden van de niveaus kan gebruik worden gemaakt van het ezelsbruggetje op de slide[1].

[1] Met dank aan Mike van Laar

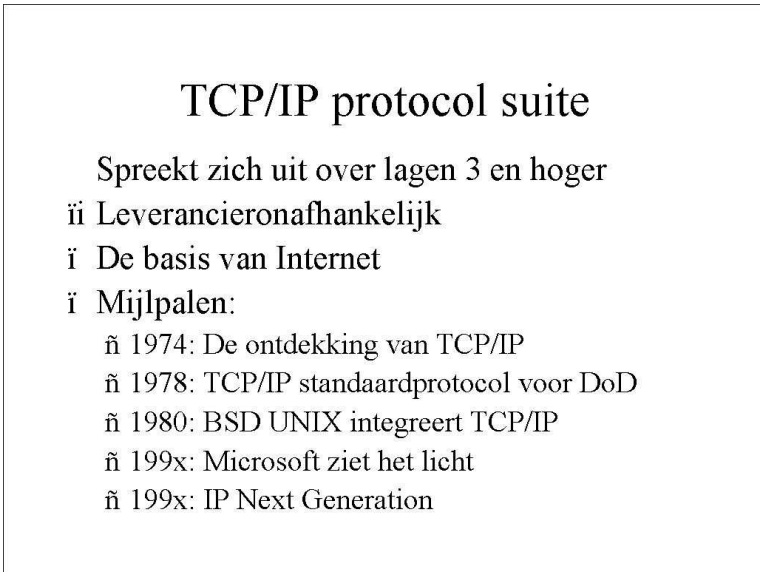
A.10 Bekende netwerkprotocollen en -architecturen



Notities

Door de jaren heen hebben een groot aantal leveranciers gewerkt aan het bieden van mogelijkheden voor het communiceren over netwerken. Het resultaat daarvan is een groot aantal onderling niet uitwisselbare netwerkprotocollen. Op het fysieke niveau is de standaardisatie een stuk beter gelukt. Transmissiemethoden als RS232, ethernet en token ring zijn wereldwijd gestandaardiseerd.

A.11 De TCP/IP protocol suite



Notities

Één van de meest populaire protocolstapels aller tijden wordt gevormd door de TCP/IP protocolsuite. Dit is een onderling samenhangende verzameling protocollen voor communicatie tussen applicaties over grote netwerken (Wide Are Networks). TCP/IP spreekt zich alleen uit over de lagen 3 en hoger van het OSI-model. Voor de data link en fysieke lagen wordt gebruik gemaakt van andere standaarden zoals RS232, ethernet, token ring en PPP. Om deze reden kan TCP/IP worden gedraaid over vrijwel elk denkbaar transmissiemedium.

Een erg in het oog springend feit is dat TCP/IP niet is ontwikkeld door een leverancier, maar door een research projectgroep. Dientengevolge is TCP/IP public domain en kan het door iedereen worden geımplementeerd zonder auteursrechtelijke of octrooi consequenties. Alle zichzelf respecterende leveranciers hebben TCP/IP daarom geımplementeerd en het is onder andere deze leverancieronafhankelijkheid die ervoor heeft gezorgd dat TCP/IP de basis vormt van het wereldwijde Internet.

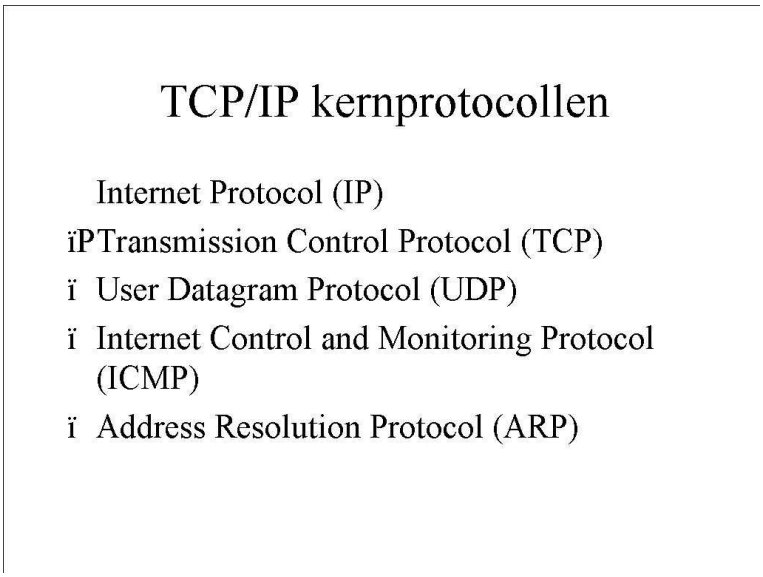
TCP/IP is omstreeks 1974 “ontdekt” door de Amerikanen Kahn en Cerf. Deze waren bezig met de ontwikkeling van een WAN-protocol wat gebruikt kon worden in het ARPANET onderzoeksnetwerk. De tot dan toe gebruikte ARPANET protocollen voldeden namelijk niet meer. TCP/IP werd een aantal malen geïmplementeerd en bleek al snel een goed werkend protocol te zijn. Het feit dat TCP/IP niet gebonden was aan één leverancier bracht het Amerikaanse ministerie van defensie (DoD) ertoe om in 1978 TCP/IP uit te roepen tot de datacommunicatiestandaard van de Amerikaanse overheid.

In een revolutionaire beweging besloot de universiteit van Berkeley om hun gratis versie van UNIX (BSD UNIX) standaard te voorzien van TCP/IP. Tot dan toe was het namelijk tamelijk ongebruikelijk om netwerksoftware te bundelen in een besturings-

systeem (is sindsdien erg gebruikelijk). Naast de kale TCP/IP communicatieprogrammatuur ontwikkelde de professoren, docenten en studenten van de Berkeley universiteit ook allerlei programmatuur die van TCP/IP gebruik maakte. Vrijwel al die programmatuur wordt in het hedendaagse Internet nog gebruikt!

BSD UNIX kon tegen een schappelijk tarief worden gekocht (source) en vond op die wijze zijn ingang op veel hogescholen en universiteiten wereldwijd. Ook het bedrijfsleven ging (onder aanvoering van de ex-studenten, nu manager) over op UNIX en TCP/IP. Anti-TCP/IP bolwerken als IBM (SNA) en DEC (DECNET) hebben het een tijd volgehouden maar moesten na verloop van tijd ook overstag. Zelfs Microsoft ontdekte de kracht, eenvoud en schoonheid van TCP/IP en implementeerde dit protocol in zijn besturingssystemen.

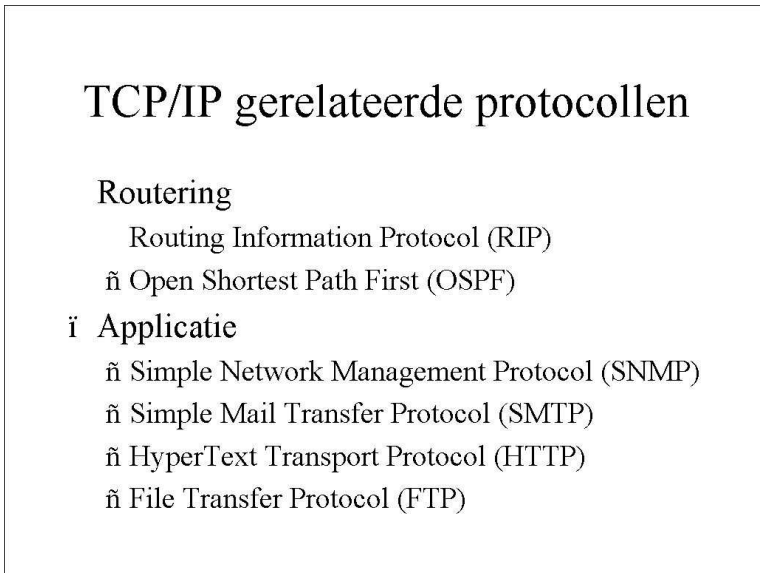
A.12 TCP/IP kernprotocollen



Notities

TCP/IP bestaat uit een groot aantal onderling samenhangende protocollen die ieder een gedeelte van de gehele netwerkproblematiek adresseren. De meest belangrijke protocollen die op iedere TCP/IP node zijn geımplementeerd zijn IP, TCP, UDP, ICMP en ARP.

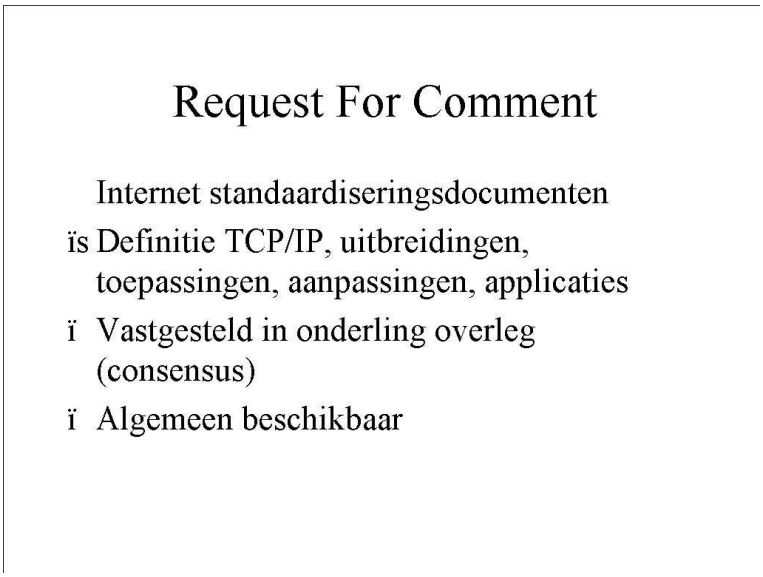
A.13 TCP/IP gerelateerde protocollen



Notities

Aan TCP/IP gerelateerde protocollen zijn applicatie of ondersteunende protocollen die extra functies bieden bij het opzetten of beheren van een TCP/IP netwerk. Op de slide staan een aantal van de meest bekende protocollen op dit gebied.

A.14 Request for Comment

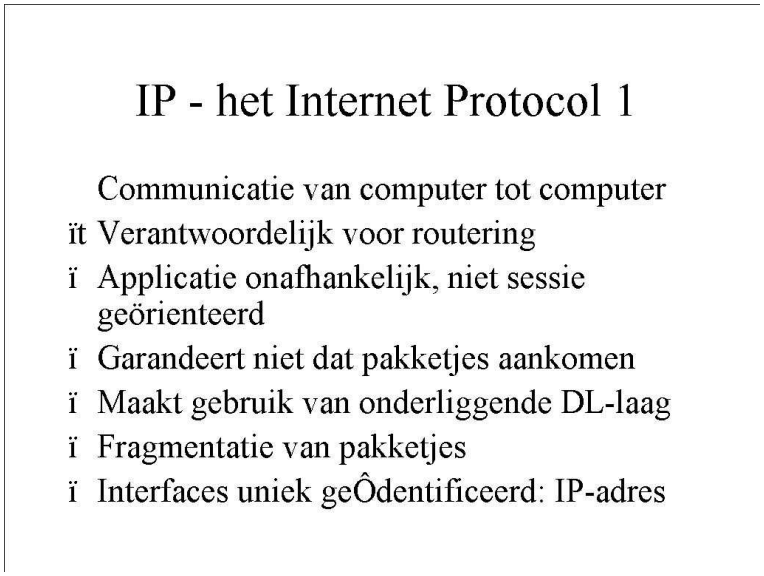


Notities

Omdat TCP/IP niet in handen is van een leverancier worden standaarden op het gebied van TCP/IP (en Internet) gezet door commissies van “vrijwilligers” onder aansturing van de Internet Architecture Board (IAB). Werkgroepen van specialisten stellen nieuwe standaarden op, verwerken commentaar en passen bestaande standaarden aan. Een nieuwe standaard wordt opgesteld als *Request for Comment*. Zolang de standaard nog de status *”Proposed standard”* heeft kan er door iedereen commentaar worden geleverd welke vervolgens in de RFC wordt verwerkt. Zodra de standaard door de IAB is geaccepteerd wordt een ieder geacht rekening te houden met de in de RFC vastgelegde wijsheden.

RFC’s zijn algemeen beschikbaar op informatie CD’s en op Internet FTP servers.

A.15 IP - Het Internet Protocol 1



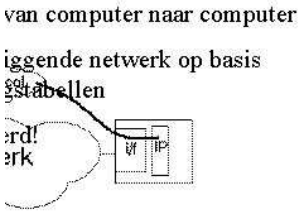
Notities

Het hart van TCP/IP wordt gevormd door het *Internet Protocol*. IP is verantwoordelijke voor de verzending van pakketjes door het netwerk. IP ontvangt deze pakketjes van de hoger gelegen lagen (4 en hoger) en stuurt ze via een bepaald netwerkinterface door naar de volgende node in de reis. Voor dit laatste maakt IP gebruik van netwerkdrivers die voldoen aan OSI laag 2 (data link)

Als IP een pakketje onvangt dan wordt gecontroleerd of dat pakketje bedoeld is voor deze node. Als dat zo is dan wordt dit pakketje doorgegeven aan de hoger gelegen netwerkprogrammatuur. Is het pakketje niet bestemd voor de huidige node dan bekijkt IP de eindbestemming en hij bedenkt waar hij dit pakketje vervolgens eens naar toe zal sturen (volgende hop). Één van IP's belangrijkste taken is dus de routering van pakketjes door het netwerk. Een pakketje hopt van node tot node door het netwerk totdat het op zijn eindbestemming is. Iedere TCP/IP node doet aan deze routeringstaak mee! De in grotere netwerken gebruikelijke routers zijn speciale computers waarin in principe alleen maar IP-software draait.

IP weet niet waarom de pakketjes worden vestuurd. Het protocol kent geen sessies of applicaties. Ieder pakketje staat op zich en wordt onafhankelijk van alle andere pakketjes verstuurd. Als een IP-pakketje zoekraakt (omdat bijvoorbeeld een router uitvalt terwijl die het pakketje aan het verwerken is) dan is dat pech. De hoger gelegen lagen moeten dit oplossen.

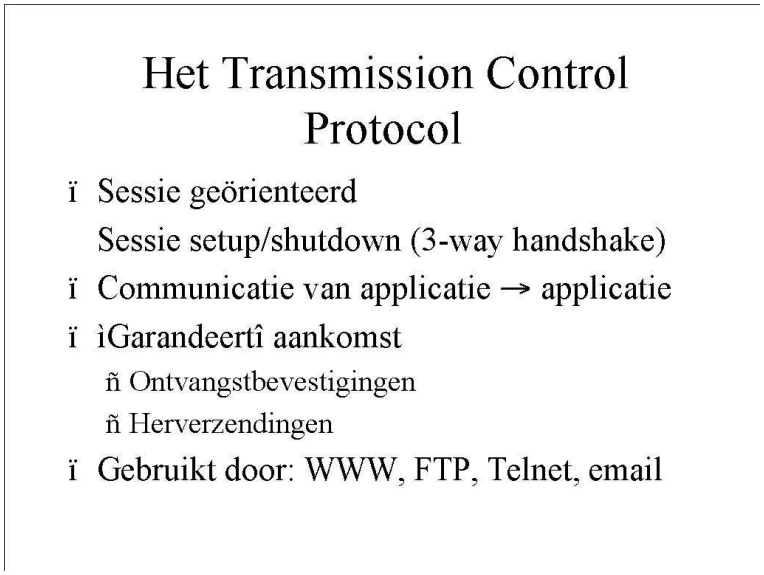
A.16 IP - Het Internet Protocol 2



Notities

Op deze slide staat met een plaatje aangegeven wat de taak van het Internet Protocol is. Zodra een pakketje op de gewenste bestemming is aangekomen wordt het daar verder verwerkt door de hoger gelegen netwerkprogrammatuur.

A.17 Het Transmission Control Protocol



Notities

IP lost een aantal belangrijke problemen in het verzenden van informatie voor ons op, maar laat evengoed een aantal andere taken liggen. Zo kunnen IP pakketjes zoekraken, in de verkeerde volgorde binnenkomen en weet IP niet voor welke applicatie in de doelcomputer een pakketje bestemd is. Het Transmission Control Protocol (TCP) lost deze problemen wel op.

TCP ondersteunt *sessies* tussen twee applicaties. Applicaties openen een TCP verbinding en gebruiken TCP-functies om informatie te verzenden en te ontvangen. De TCP laag is er vervolgens verantwoordelijk voor dat de informatie in de juiste volgorde en met de juiste inhoud aankomt. Vrijwel alle sessie geörienteerde TCP/IP applicaties (zoals email, World Wide Web, FTP) gebruiken TCP. De programmeur kan erop vertrouwen dat als hij een pakketje data aan TCP heeft aangeboden dat TCP er zorg voor draagt dat die informatie ter bestemde plekke wordt afgeleverd.

Het is dus TCP zijn taak om een betrouwbare sessieverbinding op te zetten over een onbetrouwbaar medium (IP). Om dit voor elkaar te boksen nummert TCP alle uitgaande berichten. De ontvangende TCP stuurt bevestingen van alle ontvangen pakketten. Stel nu dat TCP de pakketjes 1, 2 en 3 verstuurt. Na verloop van tijd ontvangt hij van zijn partner aan de andere kant van het netwerk de bevestiging dat de pakketjes 1 en 2 zijn ontvangen. TCP gaat er dan vanuit dan pakketje 3 zoek is geraakt en verzend dat nog een keer. Stel nu dat de verzending van pakketje 3 nu wel goed gaat. De ontvangende TCP zendt een bevestiging voor pakketje 3 naar de zender. Ook zo'n bevestiging kan echter zoekraken. In dat geval gaat de zender (time out) ervan uit dat pakketje 3 nog steeds niet is aangekomen, en zendt het dus nog een keer. De ontvanger ziet nu voor de tweede keer pakketje 3 binnenkomen, gooit het pakketje weg en stuurt wederom een bevestiging. Op deze wijze kan TCP een betrouwbaar pad tussen twee applicaties

opzetten.

A.18 Het User Datagram Protocol

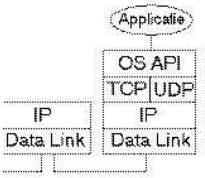
Notities

Een bekend nadeel van TCP is dat het protocol relatief veel overhead met zich meebrengt als de hoeveelheid te verzenden data gering is. Bij het opzetten en sluiten van een sessie wisselen de TCP-partners wat informatie met elkaar uit over volgnummers en het aantal pakketjes wat mag worden verzonden zonder eerst op een bevestiging te wachten (de zogenaamde *window size*.

Als we een applicatie hebben die slechts één bericht wenst te verzenden zonder antwoord van de ontvanger dan is de relatieve overhead van het opzetten en sluiten van de sessie zeer groot. Een goed voorbeeld hiervan is het HTTP protocol (World Wide Web) wat TCP verbindingen gebruikt voor het ophalen van HTML pagina's.

Speciaal voor dit soort applicaties is er het User Datagram Protocol. Dit protocol maakt telegramachtige gegevensuitwisselingen tussen applicaties mogelijk. Met UDP functies kunnen applicaties een datagram sturen naar een andere applicatie in het netwerk. UDP verpakt het datagram in één of meer IP-pakketjes en stuurt deze (via IP) naar de doelcomputer. Aangezien UDP gebruik maakt van IP zonder extra maatregelen te treffen kunnen UDP datagrammen zoekraken. Een applicatie is in dat geval genoodzaakt om zelf zorg te dragen voor herverzendingen en aanverwante verschijnselen. Het grote voordeel van UDP is de lage overhead en efficiënt gebruik van de beschikbare bandbreedte.

A.19 De informatiestroom



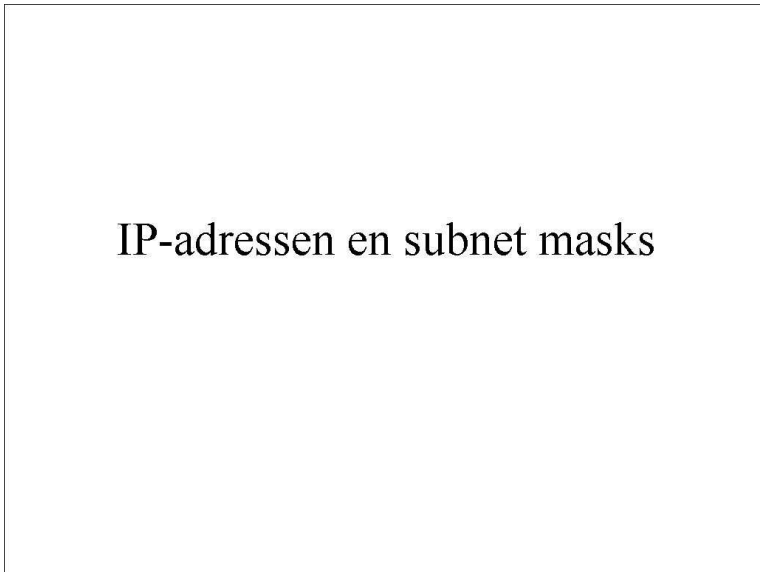
Xb∞u∞vùbaùâav∞∞∞

Notities

Op de slide is weergegeven hoe informatie van applicatie tot applicatie stroomt via de TCP/UDP/IP protocollen. Bedenk dat iedere computer in een TCP/IP netwerk in principe ook kan functioneren als router. Met name computers met meer dan één interface lenen zich goed voor deze functie. TCP/IP routers kunnen dus ook “vertalen” van het ene data link medium naar het andere. Veel routers hebben meerdere verschillende soorten interfaces. TCP/IP merkt hier verder niets van. Ieder data link medium is voor IP gelijk.

A.20 IP-adressen en subnet masks

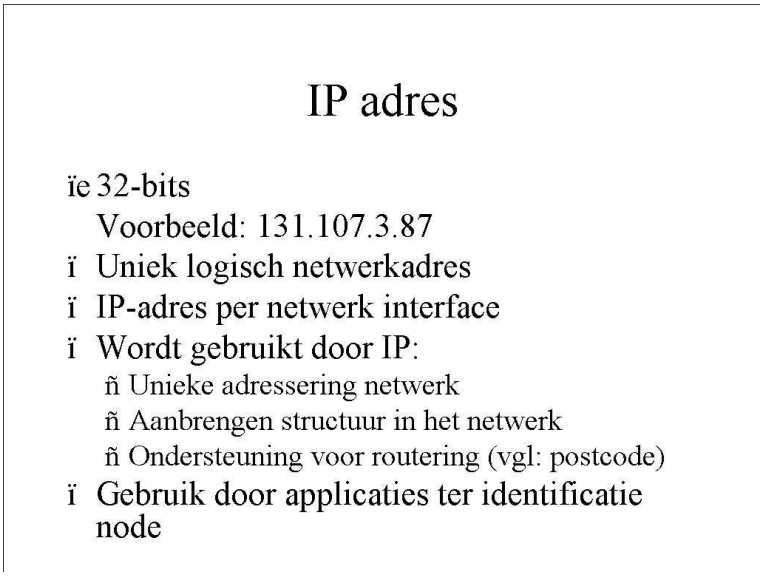
A.21 IP-adressen en subnet masks



Notities

In deze module gaan we dieper in op de adressering en naamgeving van nodes in een TCP/IP netwerk.

A.22 IP adres



Notities

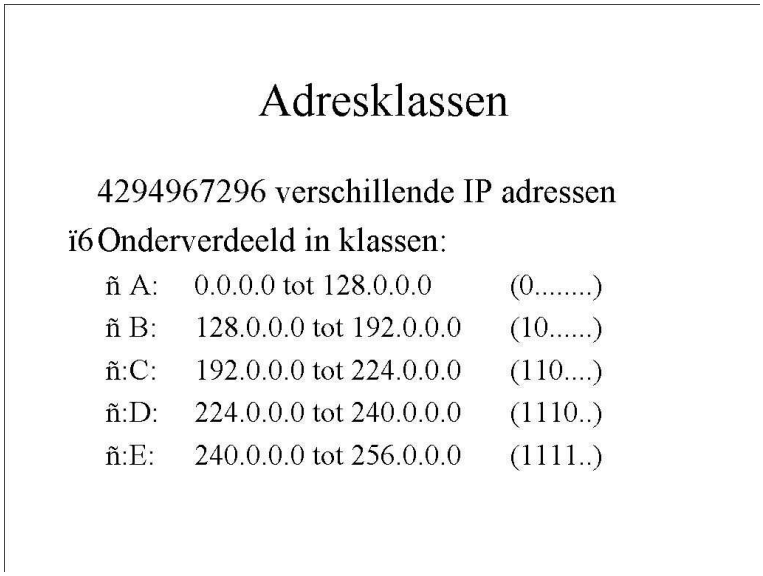
Ieder interface in een TCP/IP netwerk dient te zijn voorzien van een uniek IP-adres. Zo'n adres is een 32-bits logisch netwerknummer wat wordt toegekend door de netwerkbeheerder. Het "logische" aspect aan het IP-adres is dat het toegekende nummer niet is verbonden met het soort of type netwerkinterface aan wie het is toegekend. Een IP-adres dient in het gehele TCP/IP netwerk uniek te zijn. In Internet omgevingen houdt dit in dat toegekende IP-adressen wereldwijd uniek moeten zijn!

Uit notatieredenen schrijven we een IP-adres meestal als een combinatie van vier getallen van nul tot en met tweehonderdvijfenvijftig. Voor de wiskundigen onder ons: $256^4 = (2^8)^4 = 2^{(8*4)} = 2^{32}$.

Theoretisch kunnen er in een TCP/IP netwerk dus iets meer dan vier miljard verschillende interfaces worden opgenomen. Strikt gesproken (pyramide van Maslov) zou dit meer dan genoeg moeten zijn voor de gehele wereld. Het aantal IP-adressen begint echter al behoorlijk op te raken. De volgende generatie van IP, IP next generation, lost dit op door langere IP adressen. Om het gebruik van IP-adressen wereldwijd te reguleren is er een instantie die IP-adressen centraal registreert en uitgeeft: het InterNIC. Netwerken die zijn aangesloten op Internet mogen alleen geregistreerde IP-adressen toekennen.

IP gebruikt IP-adressen voor de interne adressering in het netwerk. Op basis van het IP-adres beslist IP of er gerouteerd moet worden (zie ook subnetting). Als er per ongeluk dubbele IP-adressen worden uitgedeeld ontstaan er netwerkproblemen in de communicatie van, naar en tussen de nodes met het conflicterende adres.

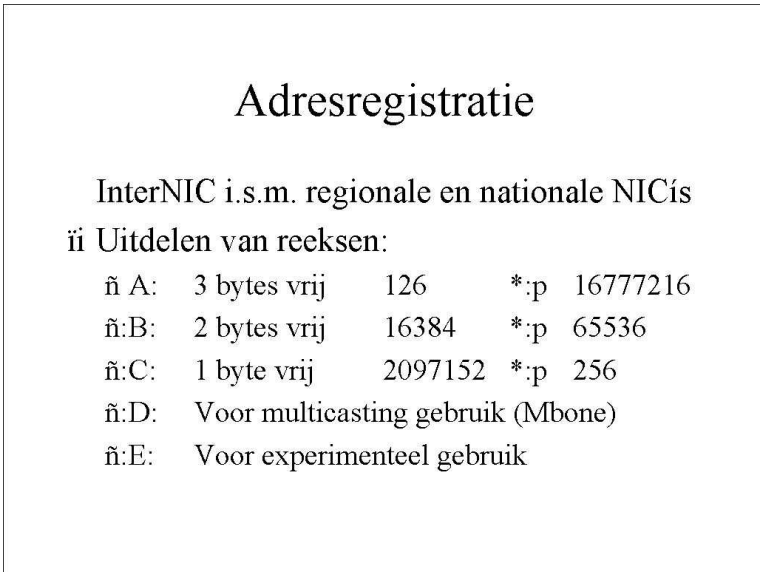
A.23 Adresklassen



Notities

Ten behoeve van het uitdelen van IP-adressen hebben de TCP/IP wijsgeren van het eerste uur besloten de gehele reeks van IP-adressen te splitsen in vijf deelreeksen (klassen).

A.24 Adresregistratie



Notities

Zoals reeds eerder gesteld verzorgt het InterNIC de registratie van IP-adressen. Organisaties die op Internet willen aansluiten dienen bij het InterNIC (of één van haar regionale zusters) een reeks van IP-adressen aan te vragen. Hoe groot de toegewezen reeks is hangt af van de grootte van het aan te sluiten netwerk en de verwachte groei over de komende jaren.

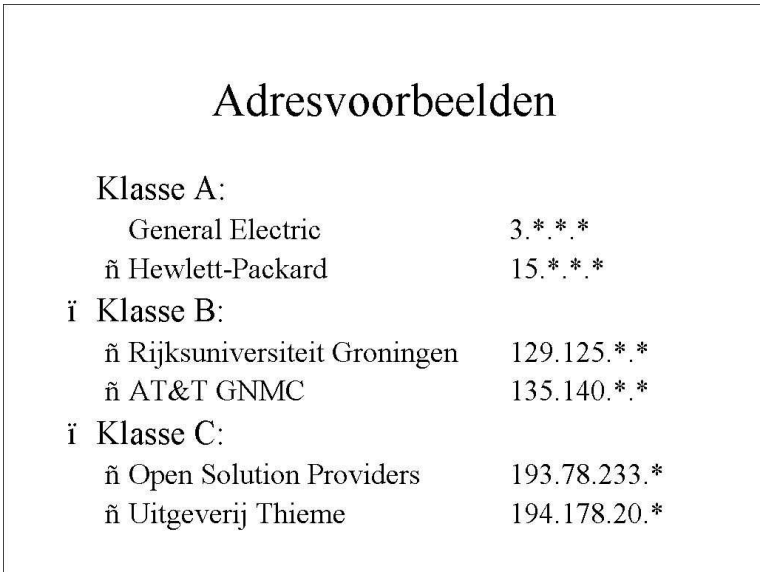
De allergrootste organisaties hebben een klasse A reeks toegewezen gekregen. Zo'n A-adresreeks heeft één byte voorgeschreven door het InterNIC en drie bytes die vrij mogen worden ingevuld. Hierdoor kunnen in zo'n netwerk zestien miljoen verschillende IP-adressen worden uitgedeeld. Wat kleinere organisaties krijgen een klasse B adresreeks toegewezen. Deze krijgen dan twee bytes voorgeschreven en mogen twee bytes zelf invullen. In deze netwerken zijn theoretisch 65536 verschillende IP-adressen te verzinnen. De kleinste netwerkjes krijgen klasse C adressen toegekend. Één byte mag worden gekozen en er worden er drie voorgeschreven. In een klasse C netwerk kunnen dus maximaal 256 verschillende IP-adressen worden bedacht. Organisaties die te groot zijn voor een klasse C reeks, maar te klein voor een klasse B, krijgen een aantal aaneengesloten klasse C adresreeksen toebedeeld.

Alle klasse A adressen zijn al vergeven. Van klasse B zijn er nog wel wat over, maar je moet van goede huize komen om er nog eentje toegewezen te krijgen. Klasse C reeksen zijn nog volop voorhande, maar het einde is niet meer onzichtbaar!

Alhoewel vier miljard+ IP-adressen op het eerste gezicht ruimschoots afdoende lijkt gaan er nogal wat IP-adressen verloren door technische oorzaken. Zo worden de klassen D en E voor andere doeleinden gebruikt. Verder mogen IP-adressen met allemaal '0'en of '1'en in het subnet of host gedeelte niet worden gebruikt. Verder moeten voor

seriële segmenten met maar twee nodes (aan de uiteinden) vaak toch een IP-adres worden toegekend. Adresschattingen van Christian Huitema doen vermoeden dat er tussen de $3 * 10^4$ en $2 * 10^8$ uitdeelbare adressen in de IP-adresreeks zitten. zijn gebleven.

A.25 Adresvoorbeelden



Notities

De uitgedeelde adressen worden bijgehouden door InterNIC en haar regionale zusters in publiek toegankelijke databases[2]. Op de slide staan voorbeelden van toegekende adresreeksen.

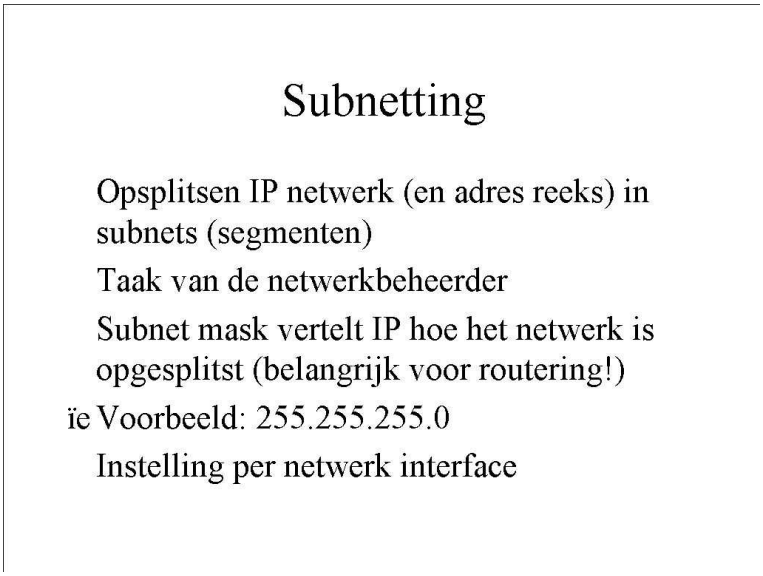
[2] telnet internic.net en telnet info.ripe.net

A.26 Configuratie IP-adres

Notities

Één van de beheertechnisch belangrijke aspecten van TCP/IP is dat iedere node in het net moet zijn geconfigureerd met een uniek, voor dat subnet van toepassing zijnd, IP-adres. Iedere PC, printer en X-terminal moet op die wijze van een IP-adres zijn voorzien. In de hedendaagse dynamische PC-netwerken levert dit aanzienlijke bezwaren op. Het beheer van uitstaande IP-adressen levert menige systeembeheerder aanzienlijke migraine aanvallen op. Microsoft heeft hierop geantwoord met DHCP, een protocol voor het automatisch toekennen en vrijgeven van IP-adressen.

A.27 Subnetting



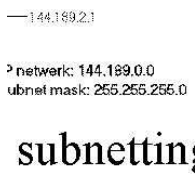
Notities

TCP/IP deelnetwerken moeten vaak ook weer worden opgesplitst in meerdere subnets. De toegewezen adresreeks moet dan weer verder worden opgesplitst om toekenning van subnet gebonden IP-adressen mogelijk te maken. De TCP/IP programmatuur in iedere node moet weten hoe het netwerk is opgesplitst in subnets. Deze kennis is van belang om te kunnen vaststellen of een andere node in hetzelfde segment (subnet) zit of dat er via TCP/IP gateways moet worden gerouteerd.

We maken de opsplitsing van het netwerk in subnets bekend door de specificatie van een *subnet mask*. Dit is een bitpatroon wat vertelt welke bits van het IP-adres behoren bij het net+subnet (de 1 bits in het masker), en welke bits horen bij het hostnummer binnen het subnet (de 0 bits). Een subnet mask van 255.255.255.0 vertelt TCP/IP dat de eerste drie bytes van het IP-adres hetzelfde zijn voor alle nodes in het lokale segment (subnet). Zodra TCP/IP zich voor de taak gesteld ziet om een pakketje te sturen naar een node met een afwijkend net+subnet id (eerste drie bytes in dit geval) dan moet het pakketje via een router reizen.

Net als het IP-adres dient het subnet mask in iedere node in een TCP/IP netwerk te worden ingesteld. Vergissingen zijn hier niet van de lucht en ook het configureren van subnet masks levert netwerkbeheerders bij tijd en wijle stevige hoofdpijnen op.

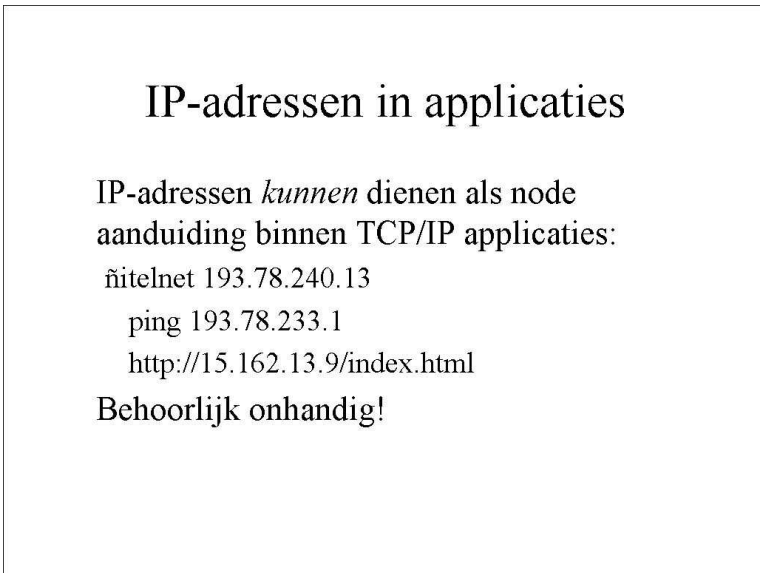
A.28 Voorbeeld subnetting



Notities

Het hele idee van IP-adressering en subnetting is wat beter te behappen met een ter zake doende voorbeeld. Zie hiervoor het netwerkdiagram op de slide.

A.29 IP-adressen in applicaties

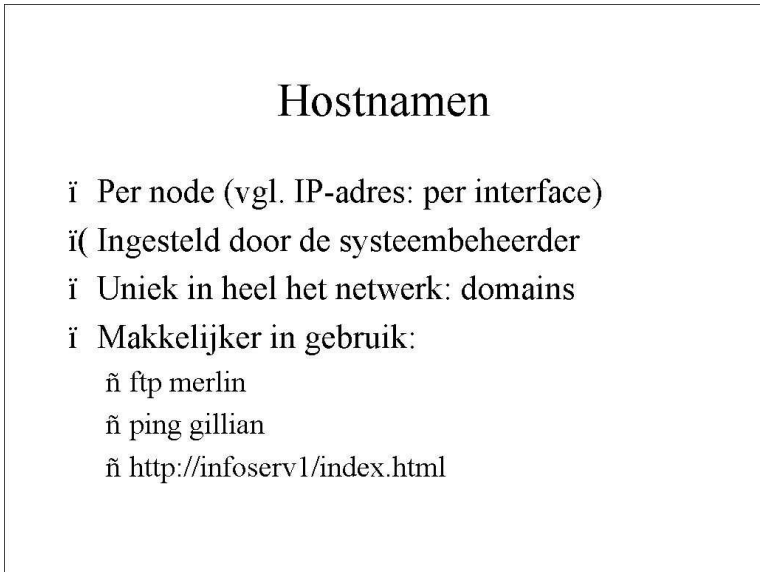


Notities

Omdat iedere node in een TCP/IP netwerk één of meer netwerkinterfaces heeft en aan een netwerkinterface één of meer unieke IP-adressen zijn gekoppeld kan een IP-adres worden gebruikt als een identificatie van een node in een TCP/IP netwerk. Om een aantal redenen verdient dit echter niet de voorkeur:

- IP-adressen zijn moeilijk te onthouden, het betreft tamelijk cryptische getallen die geen relatie hebben tot de functie van een node.
- IP-adressen kunnen wijzigen indien het deelnetwerk waarin nodes staan worden omgenummerd of opnieuw ingedeeld.

A.30 Hostnamen



Notities

De oplossing voor het op de vorige slide gepresenteerde probleem wordt geboden door het feit dat iedere node in een TCP/IP netwerk een eigen, symbolische, hostnaam heeft. De hostnaam van een systeem wordt geconfigureerd door de systeembeheerder op een voor het systeem toepasselijk wijze. Voor een UNIX systeem houdt dit in dat één of ander configuratiebestand moet worden gewijzigd, terwijl op een Windows 95 machine dit via het control panel moet worden gewijzigd.

Alle TCP/IP applicaties kunnen ook worden geparametriseerd met een hostnaam. Dus in plaats van:

```
telnet 193.78.240.13
```

kunnen we ook zeggen:

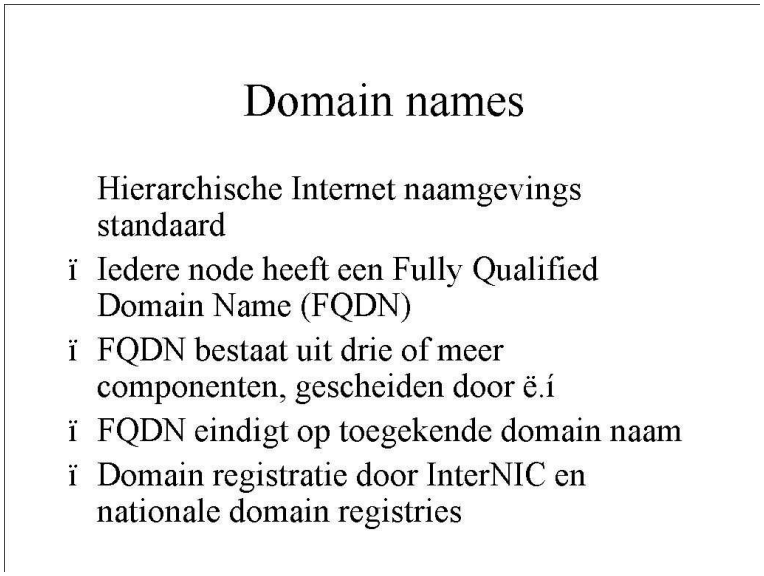
```
telnet dbserver
```

Dit laatste is natuurlijk veel eenvoudiger te onthouden. Ondanks dat we een commando parametriseren met een hostnaam werkt de TCP/IP software intern met IP-adressen. Dit betekent dat de applicatie de hostnaam moet vertalen naar een IP-adres alvorens een verbinding kan worden opgezet. Zie voor meer informatie hierover de sectie “Hostnaam resolutie” op pagina A.35.

Hostnamen dienen net als IP-adressen in het hele netwerk uniek te zijn. Binnen een deelnet is dat meestal geen probleem; de systeem- en netwerkbeheerders van de organisatie hebben de volledige controle over de naamgeving van de hosts en kunnen deze dus uniek houden. In een wereldwijd gedistribueerd netwerk als Internet is dit

natuurlijk een groter probleem. voor de hand liggende namen als ‘server’, ‘gateway’ en ‘lp’ komen natuurlijk zonder problemen in meerdere netwerken voor. De oplossing voor dit probleem wordt gevormd door hostnamen te voorzien van extra componenten waardoor ze uniek worden gemaakt. We noemen deze extra componenten de *domain names*.

A.31 Domain Names



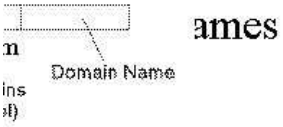
Notities

Om de uniciteit van hostnamen in het Internet te bevorderen[3] is afgesproken dat de volledige naam (de Fully Qualified Domain Name) van een node eindigt op een door de organisatie geregistreerde unieke domain naam. Doordat die domain namen wereldwijd worden uitgegeven en geregistreerd kan daardoor nooit een dubbele hostnaam voorkomen (tenzij de systeembeheerders hun werk niet goed doen).

Voor bijvoorbeeld de ANWB houdt dit in dat alle hostnamen in hun netwerken in principe eindigen op '.anwb.nl'. Aangezien geen enkele andere aangesloten organisatie die domain naam mag gebruiken is de hostnaam 'gateway.anwb.nl' wereldwijd uniek.

[3]Understatement is a form of irony in which you say *less* then you mean

A.32 Fully Qualified Domain Names



Notities

De volledig gekwalificeerde naam van een node bestaat uit *n* componenten, onderling gescheiden door een '.'. Een dergelijke naam bestaat minimaal uit drie componenten:

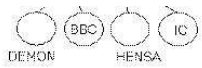
1. Simpele hostnaam (b.v. gateway)
2. Organisatie identificatie (b.v. anwb)
3. Top level domain (b.v. nl)

Organisaties mogen in principe tussen de simpele hostnaam en het verplichte gedeelte nog één of meer subdomain identificaties plaatsen. Deze dragen bij aan de FQDN. Bijvoorbeeld: 'gateway.zeeland.kantoren.anwb.nl'. Sommige landen hebben een verplicht gedeelte van drie niveaus. Bijvoorbeeld in het Verenigd Koninkrijk krijgen bedrijven een naam toegewezen die eindigt op '.co.uk'. Hierdoor is aan de FQDN te zien met wat voor soort organisatie we te maken hebben.

Domain namen worden wereldwijd gecoördineerd met het InterNIC. Per land is er vervolgens een instituut die de uitgifte voor dat land regelt. In Nederland kennen we de Stichting Domein Registratie (zie http://www.nl.net) die de uitgifte heeft uitbesteed aan KEMA.

Een spannend verschijnsel bij de uitgifte van domain namen is dat een gegeven naam maar één keer kan worden uitgegeven. In het handelsrecht van Nederland is het zo dat een bepaalde naam best twee keer mag worden gebruikt, mits de bedrijven in geheel verschillende branches opereren. Bij domain naamgeving geldt een dergelijke mogelijkheid niet. Meestal geldt hiervoor de regel: 'Wie het eerst komt, die het eerst maalt'. Verder hebben er met name in de Verenigde Staten nog wel zaken gespeeld als het registreren van merk- en handelsnamen en begrippen (www.toothpaste.com), het registreren van de handelsnaam van concurrenten en het registreren van namen van bedrijven die zo dom waren geweest zich nog niet te registreren, om vervolgens de naam door te kunnen verkopen.

A.33 Domain name hierarchie



ıe hierarchie

Notities

In de domain naamgeving die op Internet wordt gehanteerd is een strikte hierarchie aanwezig. De top level domains (laatste component in de FQDN) zijn eenduidig bepaald. Onder de top level domains hangen de organisaties die door de landelijke domain registries worden uitgegeven, weer daarondern hangen de subdomains en nodes die door de netwerkbeheerders worden bepaald.

Binnen de Verenigde Staten kunnen domains worden geregistreerd die als laatste component en aanduiding hebben van het soort organisatie:

- COM, bedrijven
- EDU, scholen en universiteiten
- ORG, stichtingen en verenigingen
- GOV, overheid
- MIL, leger
- NET, Internet gerelateerde bedrijven en instellingen

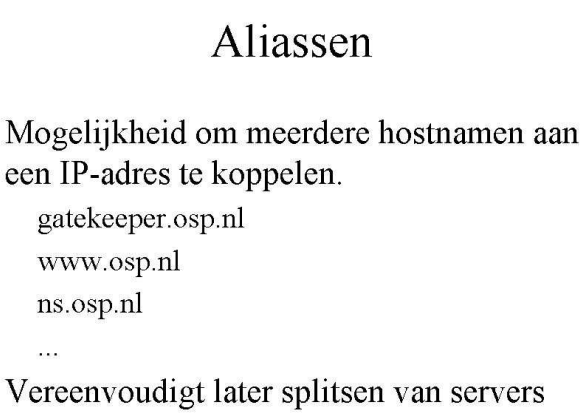
Aangesloten organisaties in andere landen krijgen een domain naam die eindigt op de ISO landcode:

- NL, Nederland
- ES, Spanje
- AU, Australië
- CH, Zwitserland
- AT, Oostenrijk
- ...

In tegenstelling tot wat veel mensen denken is er ook een .US domein. Dit wordt gebruikt door hele kleine organisaties en particulieren die een domein willen registreren. Het .US domein is geografisch ingedeeld; bijvoorbeeld '.SF.CA.US' (San Francisco, Californië).

Recentelijk zijn er enkele bewegingen op gang gekomen om het aantal toplevel domains uit te breiden met qualifiers als ".sex", ".recën ".humor". Dit vereist echter wereldwijde coördinatie omdat alle root domain name servers de nieuwe domains moeten erkennen.

A.34 Aliassen



Notities

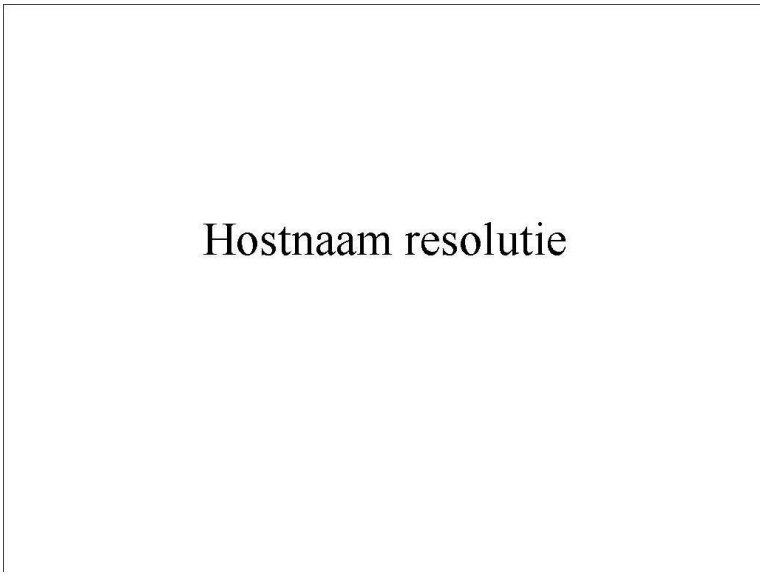
De diverse methoden voor hostnaam resolutie (zie volgende sectie) ondersteunen allemaal de mogelijkheid van het opnemen van één of meerdere aliassen voor een host. Zo is de machine ‘gatekeeper.osp.nl’ ook te benaderen als:

- www.osp.nl
- ftp.osp.nl
- mail.osp.nl
- ns.osp.nl
- news.osp.nl
- osp.nl

In dit geval wordt simpelweg hetzelfde IP-adres aan meerdere hostnamen gekoppeld. Hiervoor wordt vaak gekozen omdat het in Internet gebruikelijk is geworden om de hostnaam van een machine gedeeltelijk af te laten hangen van de service die de machine verleent. Web servers zijn bijvoorbeeld vrijwel altijd bekend als ‘www.nogwat’, terwijl FTP servers vaak ‘ftp.nogwat’ heten. Als één machine meerdere functies verricht moet hij conform deze “standaard” ook onder meerdere namen bereikbaar zijn.

A.35 Hostnaam resolutie

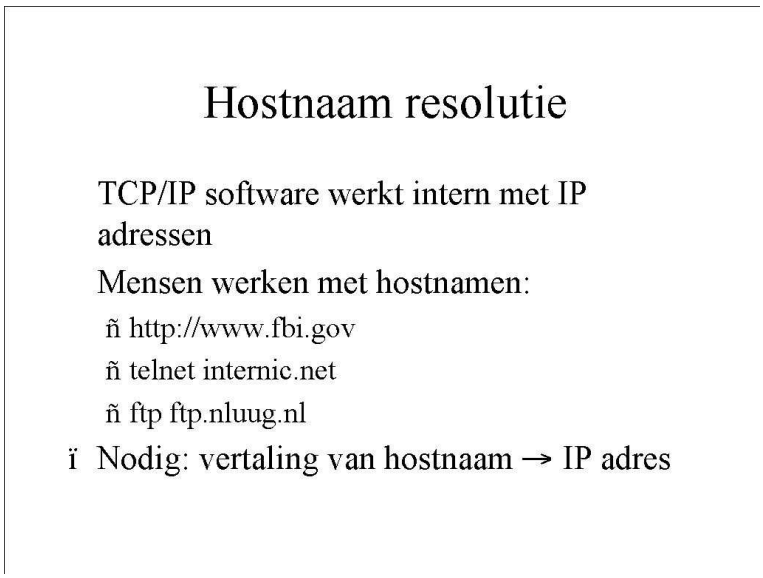
A.36 Hostnaam resolutie



Notities

Welkom bij de sectie over hostnaam resolutie. In deze sectie behandelen we de meest voor de hand liggende manieren waarop hostnamen kunnen worden vertaald naar IP-adressen en omgekeerd.

A.37 Hostnaam resolutie



Notities

Zoals reeds eerder in de shop aan de orde is gekomen werkt de TCP/IP programmatuur intern met IP-adressen, terwijl applicaties en mensen liever werken met hostnamen. Bij het opzetten van een verbinding moet een applicatie echter een IP-adres specificeren en als een applicatie informatie opvraagt over een verbinding dan krijgt deze van de TCP/IP laag weer IP-adressen terug. Er is dus behoefte aan een vertaalmechanisme van hostnamen naar IP-adressen en vice versa.

Technisch gesproken wordt deze vertaling uitgevoerd door de ‘gethostbyname()’ en ‘gethostbyaddr()’ calls. Deze UNIX en Windows systeemaanroepen voeren de gewenste vertalingen uit.

A.38 Methoden van hostnaam resolutie

Notities

In principe is iedere leverancier vrij om een mechanisme te verzinnen waarmee de vertaling kan worden uitgevoerd. Historisch en praktisch gezien worden er echter meestal één of meer van de volgende drie methoden toegepast:

1. De hosts file
2. NIS
3. De Domain Name Server

Een aantal leveranciers (zoals Microsoft) hebben hiernaast nog hun eigen methoden. Zo kan een Windows machine ook nog gebruik maken van Netbios Name Resolution mechanisme zoals local broadcast, de LMHOSTS file en de NetBios Name Server (WINS).

A.39 Hosts file

Notities

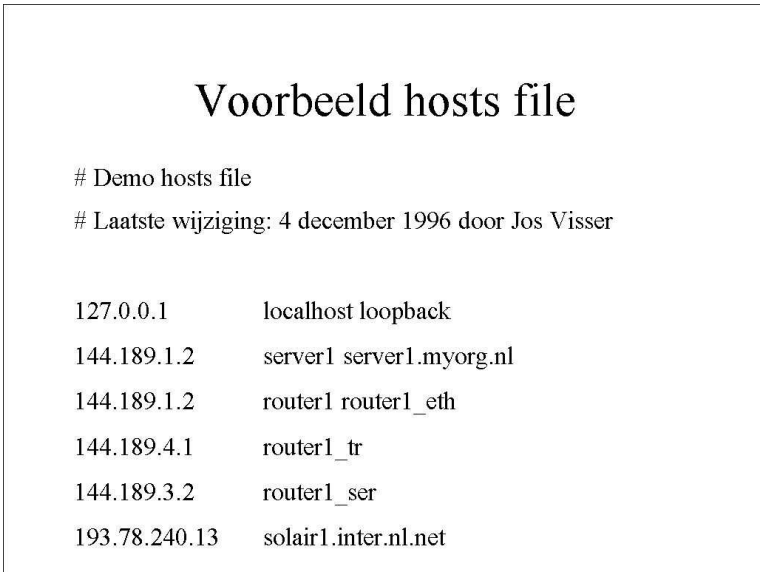
De hosts file is een tekst bestand met daarin regels met op iedere regel (afgezien van lege regels en commentaar) een IP-adres en één of meer hostnamen. Het bestand heeft zijn historische oorsprong in de file 'HOSTS.TXT'. Deze file bevatte in de vroege dagen van Internet een opsomming van alle nodes in het net met hun netwerkadressen. Vanzelfsprekend is een dergelijke aanpak vandaag aan de dag niet langer haalbaar.

De naam en plaats van de hosts file wordt door de leverancier bepaald. In UNIX systemen staat de file altijd in de /etc directory. Onderhoud op de hosts file geschiedt met een gewone ASCII editor[4]. Het grote voordeel van de hosts file is de ongekende simpelheid. Om de hosts file te gebruiken hoeft niets te worden geconfigureerd. Het onderhoud van de file is op zich redelijk simpel. De hosts file kent echter ook belangrijke nadelen:

- De file moet op iedere node in het netwerk aanwezig zijn. Dit kan leiden tot grote problemen in het distribueren van de correcte file. Met name in omgevingen waar regelmatig wijzigingen in de hosts file nodig blijk zwerven al gauw verschillende versies van de hosts file door het netwerk.
- De koppeling tussen IP-adressen en hostnamen is 1:n, terwijl de eigenlijke koppeling n:1 of n:m is. In het geval een node meerdere interfaces, en dus meerdere IP-adressen, heeft kan toch slechts één IP-adres aan een hostnaam worden gekoppeld (zie voorbeeld volgende slide).
- De hosts file kent geen faciliteiten voor het opslaan van andere informatie zoals email gateways.

[4] vi rules!

A.40 Voorbeeld hosts file



Notities

Op de slide staat een voorbeeld van een hosts file. Zoals te zien is betreft het een vrij recht-toe-recht-aan tekstbestand waarin de koppelingen tussen hostnamen en IP-adressen wordt gelegd.

Nadere analyse van de hosts file leidt ons tot de gedachte dat er een router is met de naam ‘router1’ met drie verschillende netwerk interfaces. Deze interfaces hebben de IP-adressen 144.189.1.2, 144.189.4.1 en 144.189.3.2. Het hosts file mechanisme staat ons niet toe om aan één hostnaam meerdere IP-adressen te hangen. We zijn dus genoodzaakt om aliassen op te nemen voor de verschillende netwerk interfaces: ‘router1_eth’, ‘router1_tr’ en ‘router1_ser’. Een probleem daarbij is dat applicaties die met een bepaalde hostnaam werken (bijvoorbeeld ‘router1’) altijd met hetzelfde IP-adres (netwerkinterface) praten. Afhankelijk van hoe intelligent de routering is opgezet vormt dit geen, een klein of een groot probleem.

A.41 Network Information Service

Network Information Service
(NIS)

Voorheen: Yellow Pages
Sun Microsystems
Centrale hosts tabel op NIS server(s)
ïr NIS-client doet query in tabel via netwerk
ïa Onderhoudsvriendelijker
Met name in UNIX omgevingen

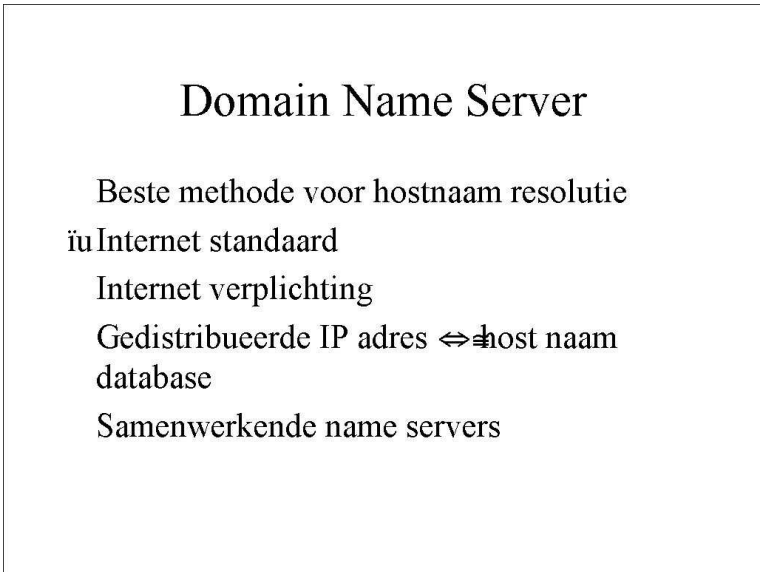
Notities

De tweede mogelijkheid voor het vertalen van hostnamen naar IP-adressen wordt gevormd door de Network Information Service (NIS). Dit is een door Sun Microsystems bedacht subsysteem wat in staat is om systeemconfiguratietabellen centraal op een netwerkserver te beleggen.

Werken met NIS is conceptueel gelijk aan het werken met een hosts file. Alleen ligt de hosts file nu niet op het lokale systeem maar op de NIS server. Vrijwel alle nadelen van de hosts file gaan ook op voor hostnaam resolutie via NIS. Het enige voordeel is dat het onderhoud van de tabel maar op één plek hoeft te geschieden, namelijk op de NIS server. Daar staat dan weer tegenover dat er een extra schakel is die kapot kan gaan, namelijk de netwerkverbinding tussen de NIS client en de NIS server.

Een operating systeem of TCP/IP protocolstapel moet expliciet ondersteuning bieden voor NIS om hiervan gebruik te kunnen maken. De in de PC-wereld alomtegenwoordige Microsoft TCP/IP software biedt die ondersteuning niet! NIS heeft dan ook vrijwel alleen opgang gemaakt in de UNIX wereld. Daarnaast is standaard-NIS ook een beveiligingsrisico van de eerste orde.

A.42 Domain Name Server

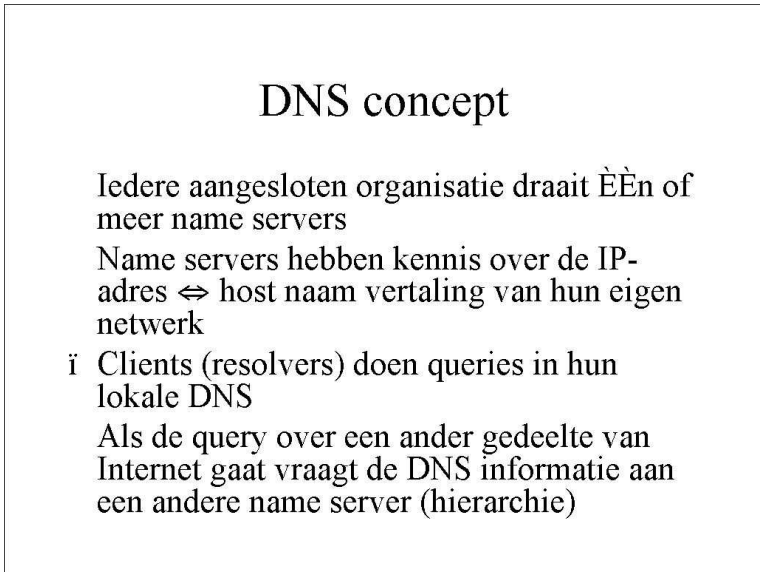


Notities

De derde methode van hostnaam resolutie is werken met een Domain Name Server. Een Domain Name Server is een serverapplicatie met kennis over de relatie tussen de hostnamen en IP-adressen van een gedeelte van het netwerk. De name server accepteert queries van klanten (de zogenaamde resolvers) en voert die queries vervolgens uit. Een resolver die wil weten welke IP-adressen er bij een hostnaam horen, of welke hostnaam er bij een IP-adres hoort, kan dit dus aan zijn name server vragen.

Tot op dit punt lijkt de werking van de DNS vrijwel gelijk aan NIS: een klant heeft een vraag over hostnaam¡-¿IP-adres vertaling en stuurt die door naar zijn lokale server. Een belangrijk verschil tussen de DNS en NIS is echter dat de DNS een query over een gedeelte van het netwerk waar hij geen verstand van heeft doorstuurt naar een andere (hogere) name server. Één name server hoeft dus niet alle hostnaam¡-¿IP-adres relaties in het gehele netwerk te weten. Alle DNS'sen samen vormen een soort gedistribueerde database waarmee de hostnaam naar IP-adres vertalingen (en vice versa) van het gehele netwerk kunnen worden opgelost. Om deze reden is de DNS het ideale mechanisme voor gebruik in Internet. Het zou immers onmogelijk zijn om een vertaaltabel (à la NIS of de hosts file) te maken voor het gehele Internet. Het gebruik van een name server is dan ook verplicht voor de aan Internet aangelosten netwerken.

A.43 DNS concept



Notities

De werking van de Domain Name Server is sterk gebaseerd op de domain naamgeving die in Internet wordt gehanteerd. Voor ieder (sub)domain geldt dat er een name server moet zijn die *authoritative* moet zijn voor dat (sub)domain. Een authoritative name server heeft de ultieme kennis op het gebied van hostnamen en IP-adressen voor dat gedeelte van het netwerk. Stel dat het bekende bedrijf Muppet Laboratories zich aansluit op het Internet dan wordt van ze verwacht dat ze een name server inrichten dir authoritative is voor het domain '.muplab.com'.

Nu kunnen de netwerkbeheerders van Muppet Laboratories ervoor gekozen hebben om het domain verder onder te verdelen in subdomains, bijvoorbeeld '.sales.muplab.com', '.research.muplab.com' en '.swamp.muplab.com'. Ook voor deze subdomains dient er een authoritative name server te zijn. Dit mag dezelfde name server zijn als de authoritative name server voor '.muplab.com'. Een name server mag namelijk autoritative zijn voor meerdere (sub)domains. Het is echter ook mogelijk om aparte authoritative name servers in te richten voor de subdomains. In dat geval zeggen we dat de autoriteit voor de subdomains is *gedelegeerd*.

Een client computer in het Muppet Laboratories netwerk die een vraag heeft over hostnamen en/of IP-adressen richt die vraag aan zijn lokale name server. Deze name server kijkt of de vraag over een (sub)domain gaat waarover hij de autoriteit heeft. Als dat zo is beantwoord hij de vraag uit zijn eigen tabellen. Als de vraag over een gedeelte van het netwerk gaat waarover deze name server geen kennis heeft stuurt hij de vraag op naar een andere name server. In de configuratie van de name server kan worden aangegeven welke name servers verantwoordelijk zijn voor welk gedeelte van het netwerk. Indien voor het gevraagde (sub)domain niet expliciet is aangegeven welke name

server ervoor verantwoordelijk is wordt de vraag doorgestuurd naar een zogenaamde *root name server*. Deze root name server kennen de name servers voor de Internet top level domains (.nl, .com, .org, ...).

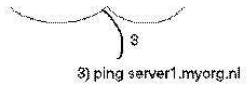
Indien een query is doorgegeven aan een andere name server kunnen er twee soorten antwoorden komen:

1. Het antwoord
   In dit geval zijn we blij. De name server aan wie we de query hebben doorgegeven wist klaarblijkelijk het juiste antwoord. Onze name server geeft nu dit antwoord terug aan de client (resolver).

2. Het IP-adres van een andere name server
   De name server wist het antwoord op de vraag niet, maar hij kent wel een name server waarvan hij vermoedt dat die het antwoord wel weet. In dat geval zal onze name server de query nog eens herhalen, maar nu naar de ‘nieuwe’ name server.

Let op dat een name server nooit een query forward over een (sub)domain waarvoor hij zelf authoritative is.

A.44 DNS voorbeeld



oorbeeld

Notities

In dit voorbeeld is grafisch weergegeven hoe een DNS query in zijn werk gaat.

A.45 Resolver configuratie voorbeeld

Notities

In een netwerk waarin gebruik wordt gemaakt van een Domain Name Server dienen een aantal zaken te zijn geregeld. Allereerst dienen er natuurlijk één of meer systemen als DNS te zijn ingericht. Deze server systemen moeten 7x24 uur bereikbaar zijn voor het beantwoorden van name queries. Dit laatste geldt met name indien het netwerk is aangesloten op het Internet. Domain Name Servers worden daarom meestal geımplementeerd op de UNIX of Windows/NT systemen die ook andere Internet server taken verrichten (zoals email server en World Wide Web server).

De configuratie van een DNS is tamelijk complex. Naast de wat meer voor de hand liggende vertalingen van hostnaam naar IP-adres moet een DNS ook antwoorden kunnen geven op omgekeerde queries (IP-adres naar hostnaam), of queries voor email servers (de zogenaamde MX records).

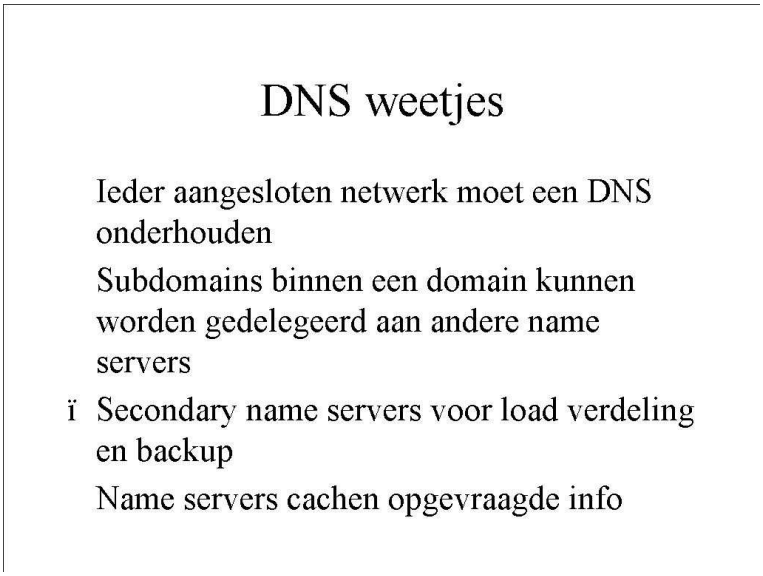
De DNS clients (de zogenaamde resolvers) zijn wat eenvoudiger te configureren. Op

de slide ziet u een voorbeeld van de configuratie van een Windows NT machine die moet worden geconfigureerd als DNS client. In principe zijn minimaal drie instellingen nodig om dit succesvol aan de praat te krijgen:

1. De (simpele) hostnaam van het eigen systeem
2. Het (sub)domain waarin deze node ‘zit’
3. Een lijst van name servers waarvan deze node gebruik kan maken

Het tweede item wordt door de resolver gebruikt om een simpele hostnaam die door de gebruiker wordt opgegeven uit te breiden met een (sub)domain gedeelte.

A.46 DNS weetjes



Notities

In principe moet dus ieder netwerk wat aan Internet is aangesloten een Domain Name Server in stand houden. Door de onderlinge samenwerking tussen de name servers kan de gehele wereld weten wat de relatie tussen IP-adressen en hostnamen is in dat gedeelte van het Internet. Het is vaak raadzaam om hele grote netwerken verder op te splitsen in subdomains. In dat geval kunnen er voor die subdomains aparte name servers worden opgetrokken. Dit verdient met name om beheertechnische redenen de voorkeur. De ‘hoofd name server’ van het netwerk delegeert dan de autoriteit over dat gedeelte van het domain naar de name server van het subdomain.

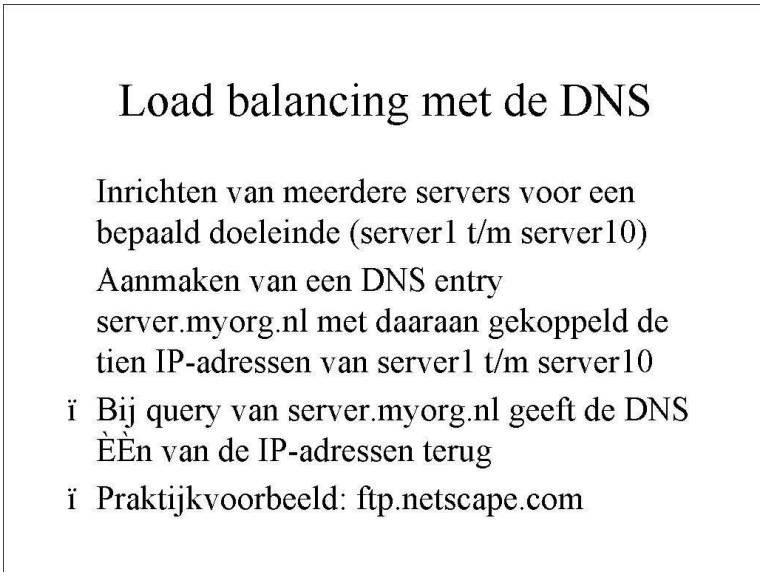
In netwerken die aan Internet zijn aangesloten heerst vaak terechte angst dat via de Domain Name Server teveel informatie ‘naar buiten’ komt over hoe het interne netwerk is opgedeeld. Beheerders hebben namelijk de mogelijkheid om via zogenaamde HINFO records extra informatie over de nodes in het netwerk op te nemen in de name server database. Deze HINFO informatie kan inbrekers op het spoor brengen van interessante systemen in het interne netwerk. Om die reden draaien veel organisaties twee primaire name servers voor het domain. Één voor intern gebruik, waarin alle informatie is opgenomen, en één voor extern gebruik, met daarin slechts een aantal (publiek toegankelijke) systemen en met beperkte informatie.

Het DNS concept kent naast primaire name servers die de ultieme kennis hebben over een (gedeelte van een) domain ook secondary name servers. Deze secondary name servers fungeren als backup name server in het geval de primaire server uitvalt en voor het verdelen van de aanvragen over de verschillende servers. Een secondary name server heeft in principe dezelfde kennis over een domain als de primaire name server. Een secondary name server hoeft echter niet te worden voorzien van deze informatie. Hij

downloadt deze automatisch van de primaire name server. Indien de tabellen van een primaire name server worden aangepast zullen de bij die primaire server behorende secondary servers automatisch (na enige tijd) een download doen van de nieuwe informatie. Door op strategische punten in het netwerk secondaire name servers te plaatsen kan de netwerkbeheerder de kwetsbaarheid en performance van het netwerk ten aanzien van name queries gunstig beınvloeden.

In een DNS omgeving is het dus in principe zo dat de resolvers (de clients) alle queries naar hun lokale name server sturen en dat die het op zich neemt om met de andere name servers op de aardbol te gaan praten over het oplossen van de query. Als de lokale name server dan uiteindelijk het antwoord op een query boven water heeft getoverd zou het natuurlijk zonde zijn indien hij dat dan onmiddelijk weer zou vergeten. Het zou namelijk best eens kunnen gebeuren dat deze of een andere client over niet al te lange tijd deze hostnaam weer eens gebruikt. Om redenen van efficiëntie *cachen* name servers daarom de antwoorden van de uitgevoerde queries. Indien een name server een query ontvangt waarvoor hij het antwoord in zijn cache heeft kan het antwoord onmiddelijk worden teruggegeven zonder dat daar verdere raadpleging van andere name servers voor hoeven te worden gedaan. Dit verhoogt de response van de name server aanzienlijk. Indien nu echter de netwerkbeheerder van de remote node besluit om de hostnaam ¡-¿ IP-adres koppelingen te wijzigen dan zijn er wellicht nog een aanzienlijk aantal name servers op de planeet die nog de oude informatie in hun caches hebben staan. Om hier enige controle over te kunnen uitoefenen kan de beheerder van een DNS voor “zijn” entries een zogenaamde *Time To Live* (TTL) specificeren. Dit is een parameter die bepaalt hoe lang de informatie in de caches van de remote name servers mag blijven staan. Name servers gooien entries uit hun cache weg waarvan de TTL is verstreken en doen indien nodig een nieuwe query bij hun collega name servers.

A.47 Load balancing met de DNS



Notities

De recente populariteit van Internet hebben de beheerders van zeer populaire sites ertoe gebracht om de verzoeken voor bestanden of HTML pagina's van die sites met behulp van de DNS op zeer creatieve wijze te verdelen over meerdere servers. Een voorbeeld van een dergelijke populaire site wordt gevormd door de FTP servers van Netscape. Het aantal verzoeken dat in piekuren aan Netscape wordt gericht is dermate groot dat het vrijwel onmmogelijk is om dit door slechts één server te laten afhandelen.

Om die reden gaan dit soort sites ertoe over om niet één maar bijvoorbeeld tien servers in te richten met precies dezelfde inhoud. Toch willen zij slechts één hostnaam aan de rest van Internet bekend maken. Via creatief gebruik van het DNS mechanisme kan ervoor worden gezorgd dat de queries voor die ene hostnaam over meerdere fysieke servers worden verspreid.

Hoe wordt dit opgelost?

Stel we richten tien servers in, met de IP-adressen 145.87.77.1 tot en met 145.87.77.10. Naar de buitenwereld willen we echter maar één hostnaam bekend maken voor gebruik door deze servers: "www.muplab.com". Wat we nu doen is dat we in de DNS van Muppet Laboratories de naam "www.muplab.com" opnemen met daaraan gekoppeld de tien IP-adressen 145.87.77.1 t/m 10. De DNS denkt nu dat www.muplab.com een machine is met tien netwerkinterface kaarten. Dat deze IP-adressen in werkelijkheid aan tien verschillende machines toebehoren weet de DNS niet.

Als een klant nu contact maakt met "www.muplab.com" wordt de name query uiteindelijk ontvangen door de DNS van Muppet Laboratories. Deze sorteert de lijst van IP-adressen in (semu-)willekeurige volgorde en beantwoordt de query. De resolver zal

normaal gesproken het eerste IP-adres in de lijst pakken en daarmee contact zoeken. Door de semi-random volgorde van de IP-adressen in het antwoord is dus nooit vooraf te zeggen met welke fysieke server er contact wordt gemaakt. Indien een groot aantal clients met www.muplab.com wil praten worden de verzoeken dus verdeeld over de tien verschillende servers van het bedrijf.

Om dit mechanisme goed te laten werken moet de DNS beheerder van Muppet Laboratories de TTL waarde van de www.muplan.com entry de waarde 0 hebben[5].

A.48 Geavanceerde onderwerpen

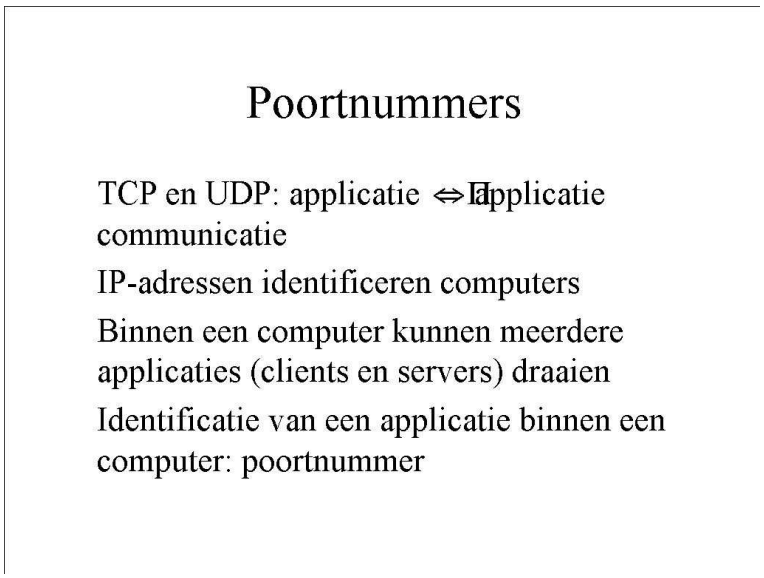
[5]The proof of this is left to the reader as an exercise

A.49 Gevanceerde onderwerpen

Notities

In deze sectie worden enkele geavanceerde onderwerpen op het gebied van TCP/IP behandeld

A.50 Poortnummers



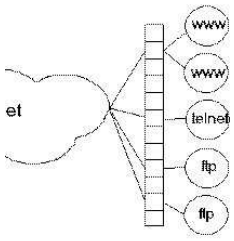
Notities

Één van de vragen die we nog niet hebben beantwoord is hoe TCP/IP bij het sturen van een pakketje naar een applicatie weet voor welke van de applicaties die er in een node draaien het pakketje bedoeld is. In principe kunnen er namelijk in een node makkelijk meerdere applicaties tegelijk draaien, en meestal zelfs meerdere *instances* van dezelfde applicatie.

De oplossing hiervoor wordt geboden door iedere TCP/IP verbinding zowel aan de client als aan de server zijde te voorzien van een uniek nummer: een zogenaamd poortnummer. Een server applicatie zal normaal gesproken bij het opstarten een TCP/IP verbinding openzetten en dan in de zogenaamde *accept* mode gaan. Dit betekent dat de applicatie stil staat totdat er door een client een poging wordt gedaan om aan die service aan te koppelen. Bij het maken van die verbinding dient de server applicatie het poortnummer wat hij wenst te gebruiken te specificeren.

Een client moet bij het opzetten van een verbinding in principe twee poortnummers specificeren: het poortnummer wat hij zelf wil gebruiken om de verbinding aan de client zijde te identificeren, en het poortnummer van de applicatie in de remote node.

A.51 Poortnummer schematisch voorbeeld



Notities

Dit concept kan worden gevisualiseerd door iedere computer te zien als een hotel met verschrikkelijk veel kamers. Op iedere kamerdeur zit een nummer geplakt, en in die kamer wonen één of meer applicaties (van hetzelfde type). Een server applicatie betrekt een kamer en hangt een bordje met “onbezet” aan de deur. Vervolgens wacht deze totdat er iemand langs komt.

Een client applicatie zoekt een vrije kamer in zijn eigen hotel en stuurt een boodschapper op pad. Deze boodschapper krijgt de opdracht om een bericht af te leveren in een bepaald hotel (IP-adres) en in een bepaalde kamer (poortnummer). De boodschapper (TCP/IP) kan aan de afzender en geadresseerde precies zien van wie het bericht afkomt en naar wie het toe moet.

A.52 Poortnummers toelichting

Notities

Met het commando “netstat -an” (UNIX en Windows/NT) kunnen we een overzicht maken van welke verbindingen er open zijn en welke servers er in de “accept” mode staan.

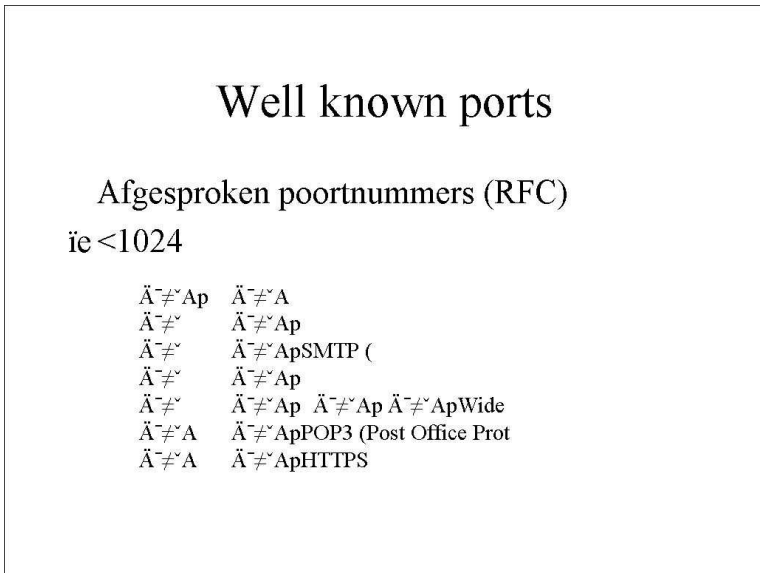
```
$ netstat -an
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:1025         127.0.0.1:1026         ESTABLISHED
  TCP    127.0.0.1:1026         127.0.0.1:1025         ESTABLISHED
  TCP    193.78.240.176:1043    193.78.233.1:23        ESTABLISHED
  UDP    0.0.0.0:135            *:*
  UDP    10.49.0.87:137         *:*
  UDP    10.49.0.87:138         *:*
  UDP    193.78.240.176:137     *:*
  UDP    193.78.240.176:138     *:*
```

Een verbinding kan uniek worden geıdentificeerd aan de combinatie van Source IP-adres+Source poortnummer en Destination IP-adres+Destination poortnummer. Aan bovenstaande output kan ik zien dat er een open verbinding is tussen een applicatie op het lokale systeem poort 1043 en een applicatie op systeem 193.78.233.1 poort 23.

Voor client systemen maakt het normaal gesproken niet uit op welke poort ze huizen. In principe kan elke poort worden gebruikt voor een uitgaande verbinding. Voor server

applicaties daarentegen is het vrijwel verplicht dat we vantevoren weten op welke poort ze zitten. Het zou anders voor een client applicatie zo goed als onmogelijk zijn om een verbinding met de server op te bouwen. Meestal is er om die reden een afspraak tussen de client en de server over welke poort er voor de communicatie wordt gebruikt. Bij een aantal TCP/IP applicaties kan de systeembeheerder bij de installatie een vrije poort selecteren en aan de applicatie toekennen.

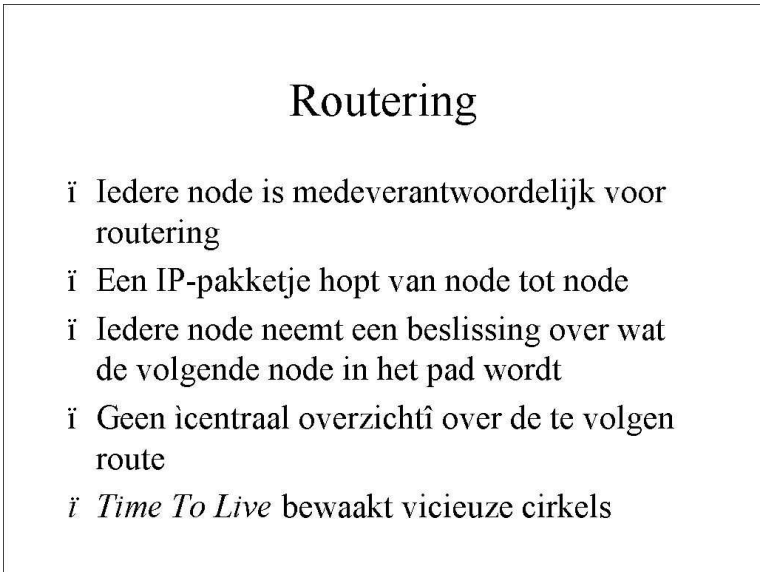
A.53 Well known ports



Notities

Om te voorkomen dat veelgebruikte applicaties ruzie krijgen over welke poortnummers ze gebruiken is er in Internet een afspraak gemaakt over welke populaire applicaties welke poortnummers gebruiken. We noemen deze de *Well Known Ports*. Zo is bijvoorbeeld afgesproken dat telnet altijd poort 23 gebruikt, het World Wide Web poort 80 en email poort 25.

A.54 Routering



Notities

Een aspect waarin TCP/IP wel tamelijk uniek is, is routering. Routering is het proces volgens welke pakketje door het complexe netwerk reizen. TCP/IP is een zogenaamd *packet switched network*. Dit betekent dat per pakketje beslissingen worden genomen omtrent hoe pakketjes door het netwerk reizen. In een TCP/IP netwerk is er geen centrale instantie die beslist hoe de pakketjes door het netwerk moeten reizen. In principe is iedere node medeverantwoordelijk voor de routering in het netwerk.

Op het moment dat een node een pakketje moet (door)zenden naar een andere node gaat hij bekijken of die andere node met één hop kan worden bereikt. Dit doet zich voor als de nodes onderdeel zijn van hetzelfde local area network of als er een rechtstreekse point-to-point verbinding tussen de nodes is. Als dat het geval is dan wordt het pakketje naar die node gezonden.

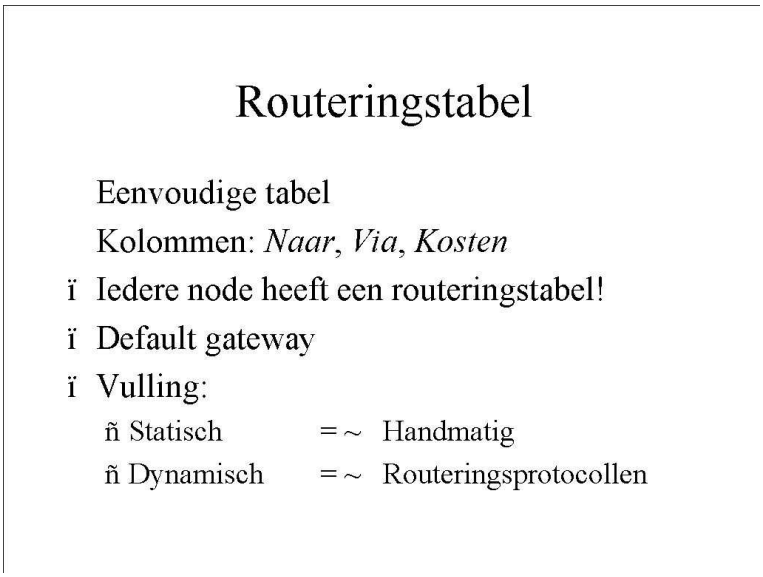
Indien de doelnode niet met één hop kan worden bereikt gaat de node zijn routeringstabellen raadplegen om te bekijken naar welke volgende node het pakketje moet worden gestuurd. Als dat kan worden vastgesteld dan wordt het pakketje bij die node afgeleverd. Vervolgens gaat die weer eenzelfde procedure in om te bekijken of het pakketje met één hop kan worden afgeleverd en zo voorts...

In principe zou zich een situatie voor kunnen doen waarbij pakketjes oneindig door het netwerk reizen zonder ooit op de plaats van bestemming aan te komen. Met name indien de routeringstabellen verkeerd zijn gevuld (bijvoorbeeld omdat ze naar elkaar wijzen) kan een dergelijke cirkel ontstaan. Om dit tegen te gaan zit er in een IP-pakketje een *Time To Live* veldje[6]. Bij iedere hop wordt dit veldje met één verlaagd. Zodra een

[6] niet te verwarren met het Time To Live veld in de Domain Name Server tabellen

node merkt dat het veldje de waarde nul krijgt, gooit deze het pakketje weg.

A.55 Routeringstabel

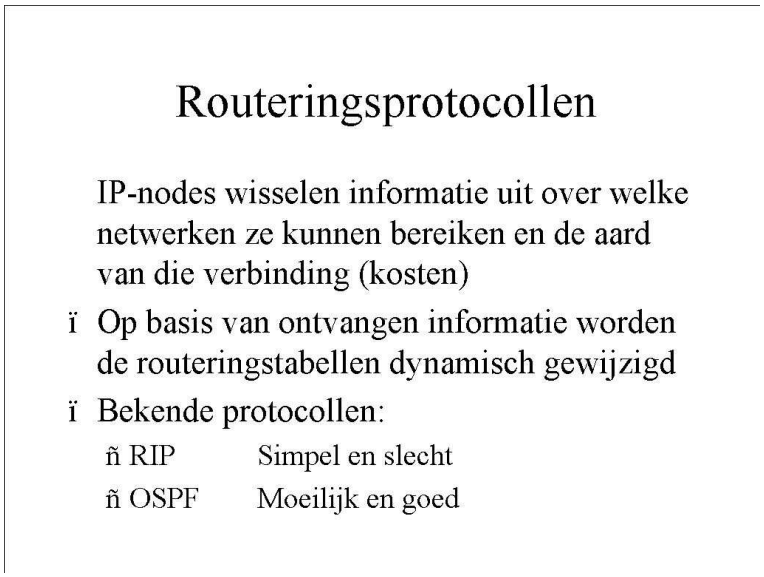


Notities

Om de routering in een TCP/IP netwerk rond te krijgen dient iedere node dus te zijn geconfigureerd met een routeringstabel. Met name de centrale nodes in het netwerk (de routers) hebben grote routeringstabellen. In principe is een IP-routeringstabel een eenvoudig ding. Het bestaat uit entries die beschrijven “Om bij *x* te komen kun je het beste het pakketje doorsturen naar *y*”. Hierbij staat *x* voor een systeem of netwerk en *y* voor het IP-adres van de node waarheen het pakketje moet worden verzonden voor de volgende hop. Aan een entry kan optioneel nog een gewicht worden gehangen. IP routeert een pakketje via de node waaraan het laagste gewicht hangt.

In veel gevallen heeft zo’n routeringstabel ook nog een *default* entry. Dit is een node waarnaartoe alle pakketje moeten worden gestuurd waarvoor geen specifiekere entry in de routeringstabel is opgenomen. We noemen dit ook wel de *default gateway*.

A.56 Routeringsprotocollen

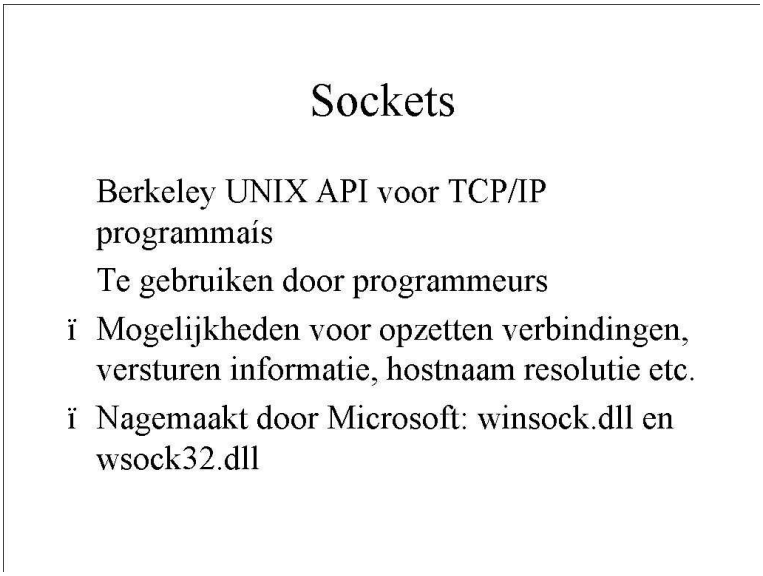


Notities

In complexe, dynamische, netwerken is het natuurlijk onbegonnen werk om de routeringstabellen van de verschillende nodes met de hand te configureren. Als verbindingen wegvallen, overbelast raken of weer *online* komen dienen de verschillende routeringstabellen te worden aangepast om deze feiten te reflecteren. Om deze dynamiek ten gunste aan te wenden zijn er door diverse bedrijven en universiteiten routeringsprotocollen bedacht. Via routeringsprotocollen houden nodes in het TCP/IP netwerk elkaar continu op de hoogte van de stand van zaken in het netwerk. Routers sturen elkaar gegevens toe over welke interfaces *up* en *down* zijn, wat de belasting van die interfaces is en welke subnetwerken via die interfaces kunnen worden bereikt. Op basis van die informatie kan een ieder zijn eigen routeringstabellen aanpassen.

Routeringsprotocollen kunnen worden beoordeeld op hoe snel en nauwkeurig ze informatie over het netwerk door het netwerk verspreiden. Twee van de meest gebruikte routeringsprotocollen van dit moment zijn RIP (Routing Information Protocol) en OSPF (Open Shortest Path First). Met name OSPF maakt de laatste tijd opgang.

A.57 Sockets



Notities

Een veelgehoorde kreet in TCP/IP land is het begrip *socket*. Wat zijn deze sockets nu eigenlijk?

Een “socket” is een programmatisch concept waarmee een programmeur een netwerkverbinding kan abstraheren. Populair gezegd vormen sockets voor de TCP/IP wereld wat *file descriptors* voor de disk wereld zijn: identificaties van open verbindingen of bestanden. In weze vormen sockets een *Application Programmer Interface* voor het openen, lezen, schrijven en sluiten van netwerkverbindingen.

De socket libraries zijn ontwikkeld door de universiteit van Berkeley als middel om vanuit een C programma TCP/IP netwerkverbindingen te manipuleren. Met de calls in de socket library kan een programmeur netwerkverbindingen opzetten, sluiten en gebruiken. Sinds de invoering zijn de Berkeley sockets de enige API voor het gebruik van TCP/IP geweest onder UNIX en VMS.

Sinds ook Microsoft het TCP/IP licht heeft zien branden en TCP/IP protocolstapels ter beschikking zijn gekomen onder DOS en Windows zijn ook de Berkeley sockets ter beschikking gekomen onder deze Microsoft besturingssystemen. De befaamde “winsock.dll” is in weze niets anders dan de van Berkeley sockets gekopieerde Microsoft standaard voor het gebruik van TCP/IP onder DOS/Windows.

A.58 IP next generation

IP next generation

IPV6

Opvolger van huidige IP (IPV4)

- Uitbreidingen op het gebied van
  - IP-adres lengte (128 bits)
  - *Class of Service*
  - Beveiliging
  - Routering
- Overgangsstrategie van IPV4 naar IPng

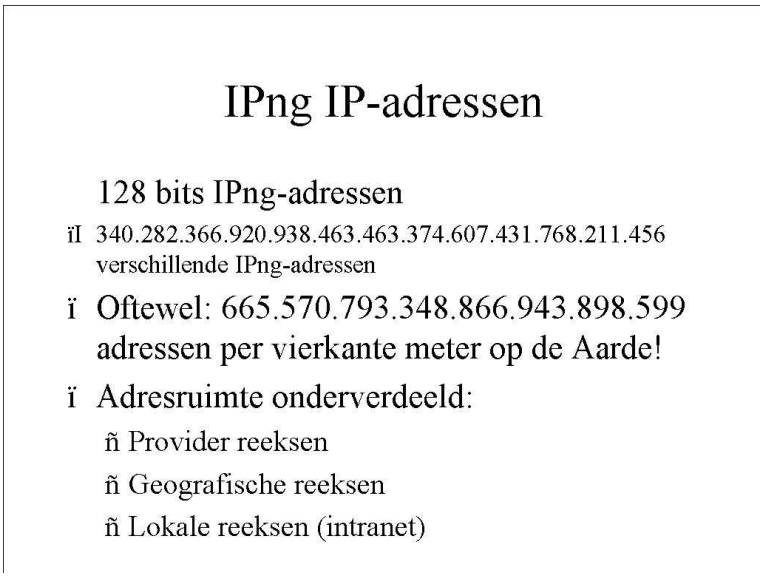
Notities

Bij tijd en wijle blijtk dat de huidige versie van IP (IP versie 4, IPV4) al weer een aantal jaartes oud is. Alhoewel IPV4 al zijn tanden nog heeft en nog leest zonder bril, blijkt toch regelmatig dat dit protocol (in computertermen) stokoud is. Om deze reden hebben de TCP/IP goeroes van het IAB de koppen bij elkaar gestoken en is er nagedacht over de volgende versie van IP.

Deze niewe versie gaat door het leven als IP next generation (IPV6, ook wel IPng). Alhoewel de wensenlijst voor uitbreiding van IPV4 tamelijk gigantisch was zijn de IAB-architecten erin geslaagd het protocol eenvoudig en simpel te houden. De meest opvallende uitbreidingen in IPng zijn de uitgebreidere IP-adressen (128 bits) en de introduktie van een *Class of Service* flow label waarmee aan een IP-pakketje een indicatie kan worden gehangen van de gewenste netwerk service (interactief verkeer, online video, file transfer, bulk verkeer en dergelijke).

Toen het ARPANET indertijd overging van de ARPANET protocollen naar TCP/IP is dit gedaan volgens het *Big Bang* model. Op uur U gingen alls computers uit en herstartten ze met de juiste netwerksoftware. Een dergelijk aanpak voor de overgang naar IPV6 is natuurlijk vandaag aan de dag uit den boze. IPV4 en IPV6 zullen geruime tijd naast elkaar moeten bestaan en er zijn dan ook allerlei voorzieningen getroffen die een geleidelijke conversie mogelijk maken.

A.59 IPng IP-adressen



Notities

Een vrij opvallende en broodnodige uitbreiding in IPng is de introduktie van langere IP-adressen: 128 bits in plaats van de 32 bits in IPV4. Dit geeft ruimte voor een werkelijk verbluffend aantal IP-adressen: namelijk 2^{128}. Volgens moderne schattingen zitten er minder dan 2^{128} atomen in heel het universum, dus het zit er wel in dat we met de 128 bits IPng adressen een tijdje vooruit kunnen.

Net als in IPV4 is in IPV6 de gehele adresreeks onderverdeeld in verschillende reeksen. Nieuw is echter dat daarin reeksen zijn onderkend voor lokaal gebruik (in een Intranet), reeksen per Internet provider en reeksen voor diverse geografische gebieden op de wereld. Deze voorzieningen bleken nodig om ook in een IPng netwerk met eindige routeringstabellen te kunnen werken.