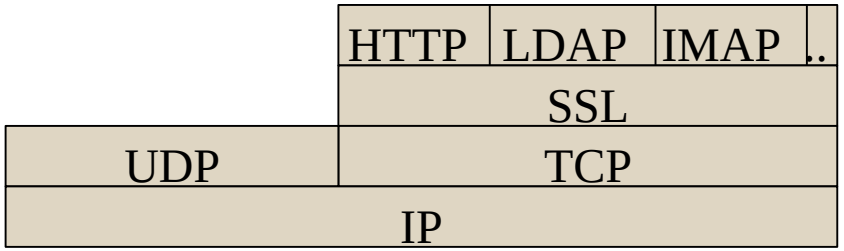
Sécurisation

SSL

SSL (1)

- SSL (Secure Socket Layer) assure :
  - l'authentification du serveur
  - la confidentialité des données
  - l'intégrité des données
  - (optionnel) l'authentification du client

- Transparent pour l'utilisateur
  - Chiffrement seulement des données
  - Se situe entre la couche TCP et la couche applicative



SSL (2)

- SSLv1 -> juillet 1994 par Netscape (jamais utilisé)

- SSLv2 -> fin 1994, intégré à Netscape Navigator
  en mars 1995 → apparition du **https**

- SSLv3 -> novembre 1995

- TLS (Transport Layer Security)
  -> normalisé par l'IETF
  -> RFC 2246, en 1999
  -> basé sur SSLv3, mais avec de petits changements, donc incompatibilité

  actuellement TLS v1.3 possible

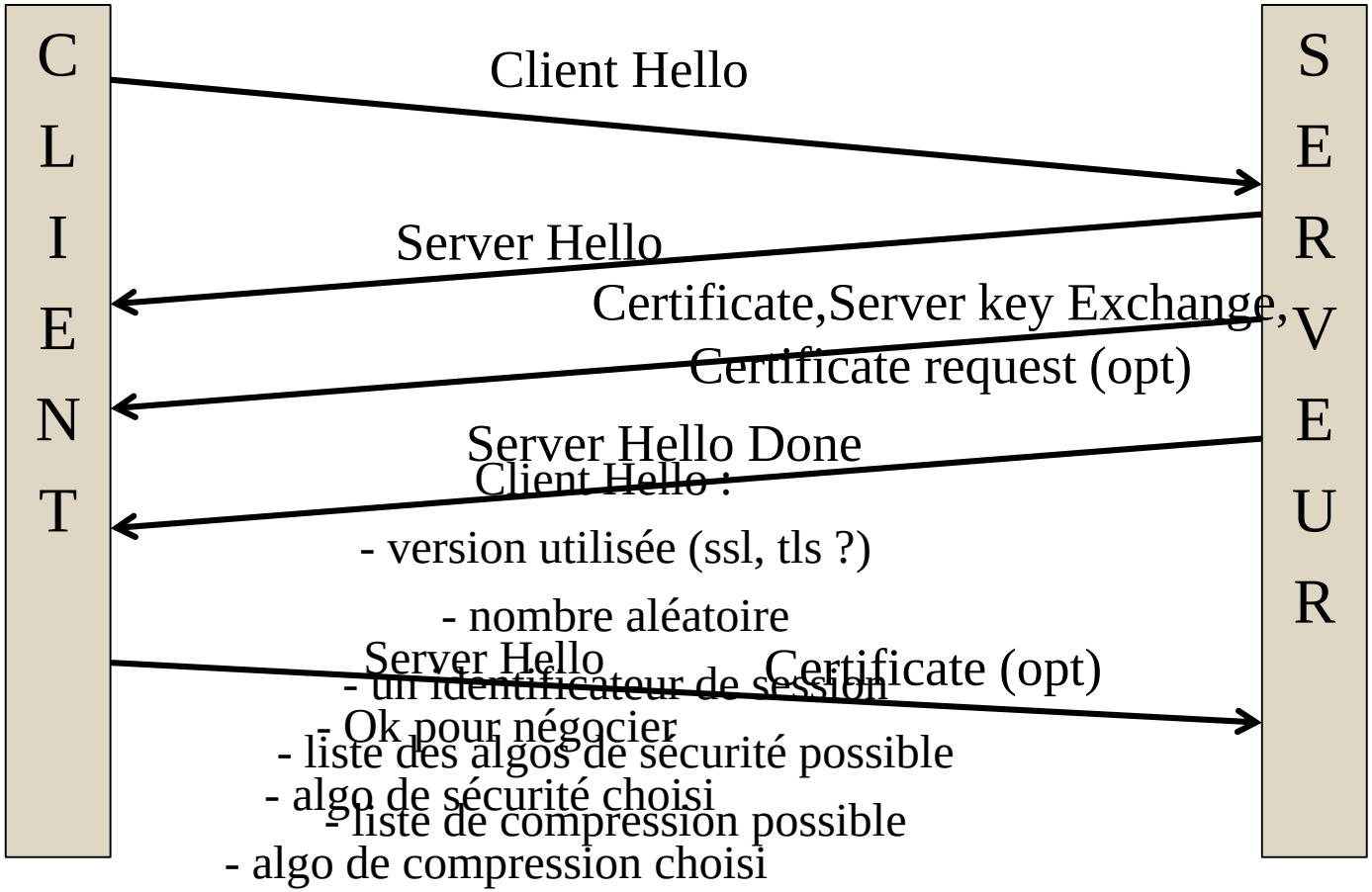
SSL (3)

- SSL est composé de deux étapes
  - SSL Handshake
    - Échange des informations pour le chiffrement (longueur de clé, protocoles utilisés, etc..)
    - Utilisation de certificats et de clés publiques pour échanger une clé de session symétrique
  - SSL Record
    - Échange des données chiffrées par la clé de session
    - Impossible à décrypter même si les échanges précédents ont été récupérés

Pré-requis serveur : Certificat + clé publique/privée

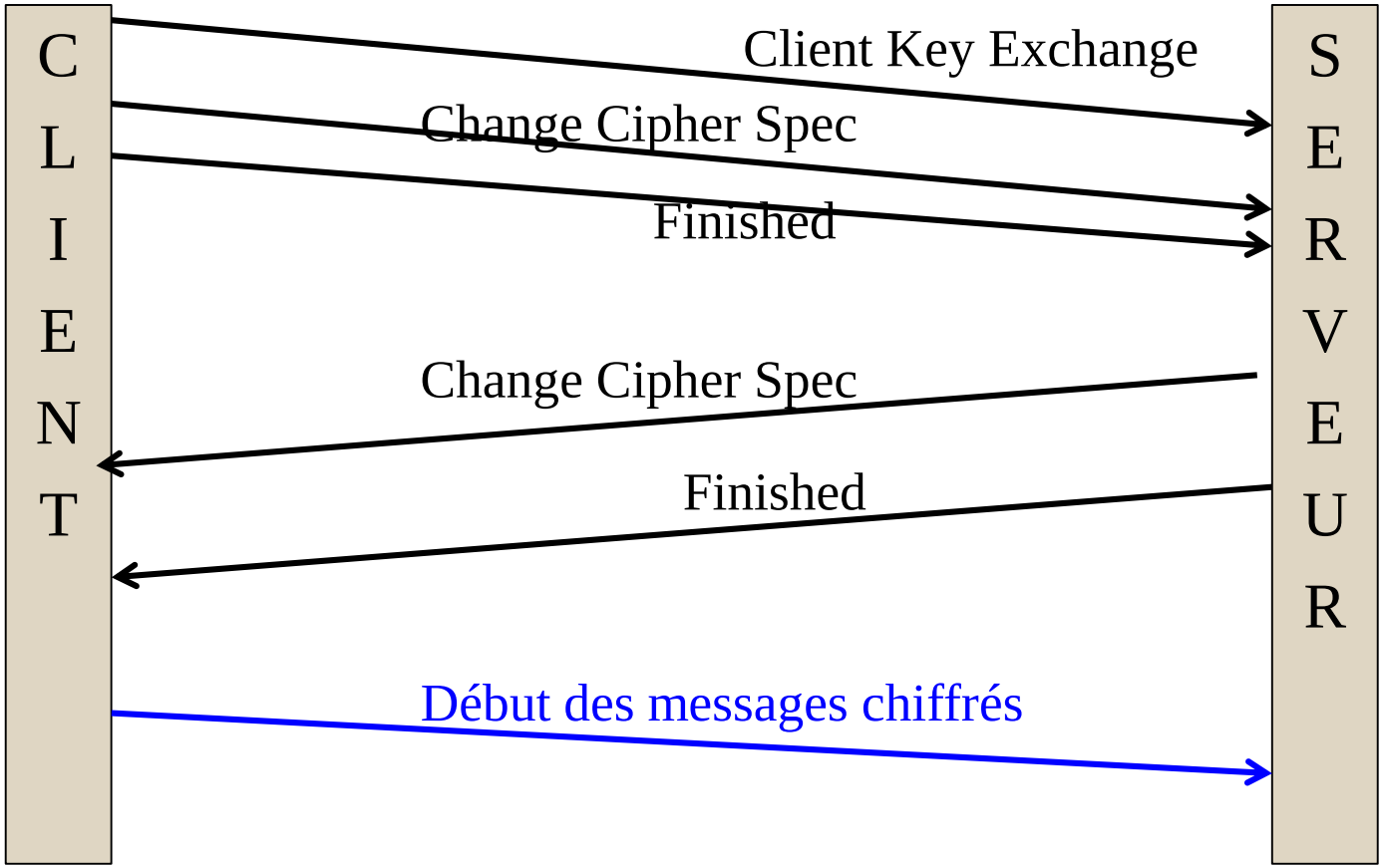
TLS 1.2 (1)

- Handshake



TLS 1.2 (2)

- Handshake (suite et fin)



Utilisation ssl en C (1)

- Ouverture de la bibliothèque, création d'un contexte
  - SSL_library_init();
  - OpenSSL_add_all_algorithms();
  - SSL_load_error_string();
- Choix d'une version de ssl
  - SSL_Method *me= SSLv3_{server/client}_method() -> que SSLv3
  - TLSv1 -> que TLSv1 , TLSv1_1 -> que TLSv1.1
  - TLS -> SSLv3, TLSv1, TLSv1.1, TSLv1.2
  - Autre méthode , déprécié (SSLv1_2)
- création d'un contexte
  - SSL_CTX *ctx=SSL_CTX_new (me)

Utilisation ssl en C (2)

- Côté serveur, chargement des certificats
  - SSL_CTX_use_certificate_file(….)
  - SSL_CTX_use_PrivateKey_file(…)
  - SSL_load_error_string();
- Création socket normal avec mise en attente de connexion
- *Mise en relation du contexte ssl avec le client*
  - ssl = SSL_new(ctx);
  - SSL_set_fd(ssl, socket de communication);
- Emission et réception
  - SSL_write(ssl,buffer, taille_buffer);
  - SSL_read(ssl, buffer, taille_buffer);

Utilisation ssl en JAVA(1)

- Utilisation des classes SSLSocket{server}Factory et SSL{Server}Socket
  - **Côté serveur**

```
SSLServerSocketFactory sslserversocketfactory =
        (SSLServerSocketFactory) SSLServerSocketFactory.getDefault();
    SSLServerSocket sslserversocket =
        (SSLServerSocket) sslserversocketfactory.createServerSocket(port);
    SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();
```

  - **Côté client**

```
SSLSocketFactory sslsocketfactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();
    SSLSocket sslsocket = (SSLSocket)
sslsocketfactory.createSocket(" nom_serveur", port);
```

Utilisation ssl en JAVA (2)

- Keystore /Truststore
  - Keystore : fichier crypté qui contient clé privé/certificat
  - Truststore : fichier crypté qui contient clé publique et les certificats

  En Java, un client se réfère toujours à un truststore

  Par défaut : *cacerts* → *sous linux /etc/pki/java/cacerts*

- Utilisation d'un autre truststore ou keystore
  - Java –Djavax.net.ssl.trustStore=montrust
    –Djavax.net.ssl.trustStorePassword=passwd nom_programme_client
  - Java –Djavax.net.ssl.keyStore=monkeystore
    –Djavax.net.ssl.keyStorePassword=passwd nom_programme_serveur

- Gestion des clés via le programme : **keytool** (inclus avec la jre)

Deux applications réseaux

Protocole HTTP(S)

SMTP

Protocole HTTP

- HTTP : HyperText Transfert Protocol
  (RFC 1945 et 2616)
  - Protocole de rapatriement des documents
  - Protocole de soumission de formulaire
  - 2 versions un peu différente : HTTP 1.0 et 1.1
    - Définition des mots-clés, et de la syntaxe
  - Fonctionnement au dessus d'une connexion en TCP
    - Utilisation d'une URI (Uniform Resource identifier)
    - Sur un navigateur
      http://<nom_machine>:<port>/<path>?<query>
  - Nouvelle version : HTTP 2.0 -> RFC 7540
    - Maintenant apparition HTTP 3.0

Méthode HTTP (1)

- Quelques méthodes

http 1.0

- **méthode GET**
  - récupération d'un document
- **méthode POST**
  - envoi de données au programme situé à l'URL spécifié
- **méthode HEAD**
  - récupération des informations sur le document (lignes d'en tête)
- PUT (document à enregistrer)
- DELETE (permet d'effacer un fichier)
- OPTIONS (différentes options utilisables par le serveur)
- CONNECT (connexion à un proxy)
- TRACE (récupérer le message reçu par le serveur)

Requête HTTP

- Une requête http comprend:
  - une ligne de requête contenant
    - la méthode ou commande
    - l'URL
    - la version du protocole utilisé (http/1.1)
  - les champs d'en-tête de la requête
    - des informations supplémentaires
      info : valeur
  - Une ligne blanche
  - le corps de la requête (si nécessaire)

  - Exemple : http(s)://www.isima.fr/index.html
    - GET /index.html HTTP/1.1\r\n
      Host: www.isima.fr\r\n
      Accept : text/html, application/xhtml+xml\r\n
      Accept-Encoding : gzip, deflate\r\n
      Connection : keep-alive\r\n
      \r\n

Réponse HTTP

- Une réponse http comprend:
  - une ligne de statut contenant
    - la version du protocole utilisé
    - le code de statut
    - la signification du code
  - les champs d'en-tête de la réponse
    - des informations supplémentaires
      info : valeur
  - Une ligne blanche
  - le corps de la réponse

```
HTTP/1.1 200 OK\r\n
Date: Tue, 22 Apr 2014 09:48:55 GMT\r\n
Server: Apache/2.4.4 (Win32) PHP/5.4.14\r\n
Last-Modified: Wed, Feb 2014 15:57:14 \r\n
Content-Length: 125\r\n
Keep-Alive: timeout=5, max=99\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html\r\n
\r\n
<html>\r\n
<head><title>page essai</title>\r\n
</head>\r\n
<body>\r\n
<H3>Vive le reseau</H3>\r\n</html>
```

Quelques en-têtes (1)

- Générique :
  - Content-length : longueur des datas
  - Content-type : type MIME des datas
  - Connection : keep-alive ou close (optimisation en http 1.1)

- Requête
  - **Host : hôte virtuel demandé (obligatoire en http 1.1)**
  - Accept : Type MIME autorisé
  - Accept-encoding : Méthode de compression autorisée
  - Cookie : cookie mémorisé par le client
  - User-agent : nom et version du logiciel client
  - If-modified-since : renvoie le document seulement si il a été modifié
  - etc

Content-length et content-type obligatoire avec requête POST

Quelques en-têtes (2)

- Réponse :
  - Date : date de la réponse
  - Server : nom et version du logiciel
  - Last-modified : date dernière modification
  - Content-language : langue utilisé pour le document
  - Expires : date à laquelle le document expire
  - Etag : numéro de version du document
  - Set-cookie : pour mettre un cookie
  - Etc

Codes de réponse

- 100 à 199
  - informationnel
- 200 à 299
  - Succès de la requête
    - 200 : OK
    - 201 : Created
    - 202 : Accepted …
- 300 à 399
  - Redirection
    - 301 : moved permanently
    - 302 : found …
- 400 à 499
  - Erreur due au client
    - 400 : bad request
    - 401 : unauthorized
    - 402 :payment required
    - 403 : forbidden
    - 404 : not found
- 500 à 599
  - Erreur due au serveur
    - 500 : internal error
    - 501 not implemented
    - 502 : bad gateway …

Commande HTTP (2)

Pour récupérer un document

- méthode GET : les paramètres sont passés dans la ligne de l'URL

  (le " " est remplacé par +, caractères spéciaux par %code ascii)

  - *ex: GET /~toto/q1.php?name1=val1&name2=val2 HTTP/1.1\r\n*
    Host : www.isima.fr\r\n
    **ligne blanche* (\r\n)*

- méthode POST permet de mettre les arguments dans le corps de la requête (→ invisible)

  - *ex : POST /~toto/q2.php HTTP/1.1\r\n*
    Host : www.isima.fr\r\n
    Content-type : application/x-www-form-urlencoded\r\n
    Content-length : xx\r\n
    ** une ligne blanche* (\r\n)*
    name1= val1&name2 = val2

Cache HTTP

- 2 caches possibles :
  - sur le navigateur
  - sur les proxys (équipement réseaux)

- En-têtes concernés
  - Last-modified
  - Expires
  - Date
  - If-modified-since
  - Cache-control
  - Etag avec if-Match ou If-None-Match

On ne renvoie pas la réponse → gain de données envoyées

Suivi de session (1)

Motivations

- la notion de session est importante dans une application conversationnelle

- Hors, HTTP est un protocole sans état

  → une connexion à chaque demande ou presque

  le serveur ne maintient pas d'informations liées aux requêtes d'un même client

- Comment implanter la notion de session sur plusieurs requêtes HTTP ?

Suivi de session (2)

- Le serveur génère un identificateur de session et associe un état à une session

  le client renvoie l'identificateur de session à chaque requête

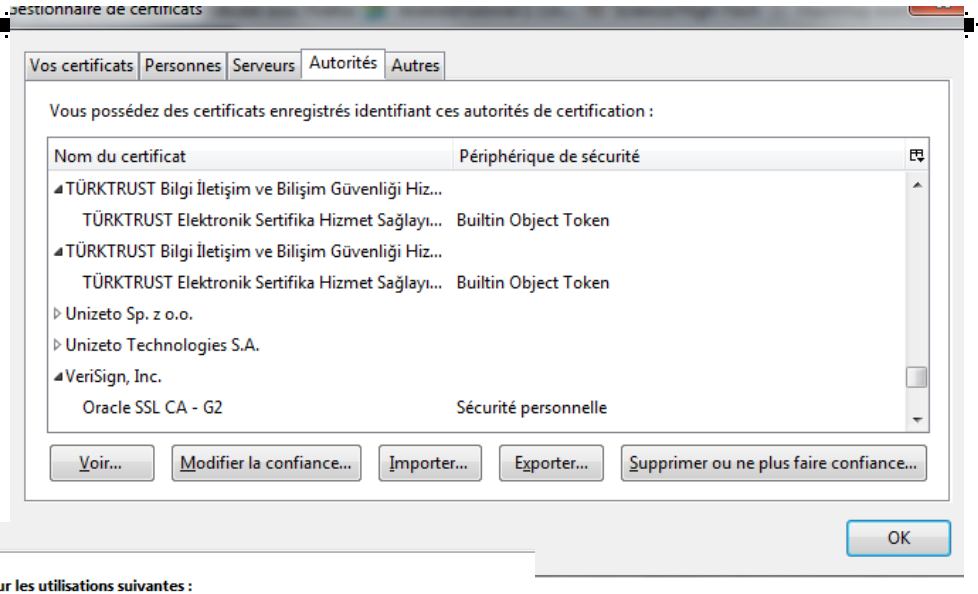
- Plusieurs solutions possibles:
  - Input HIDDEN dans les formulaires
    - on renvoie à chaque fois un identifiant
  - la Ré-écriture dans chaque URL
    - on remet l'identifiant dans les réponses
  - Utilisation d'un cookie
    - Set-cookie venant d'un serveur
    - Cookie à partir du client

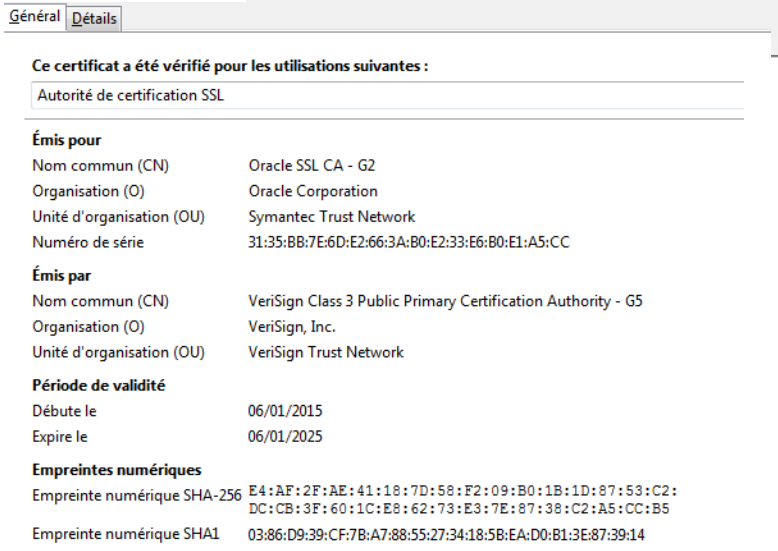
Https (1)

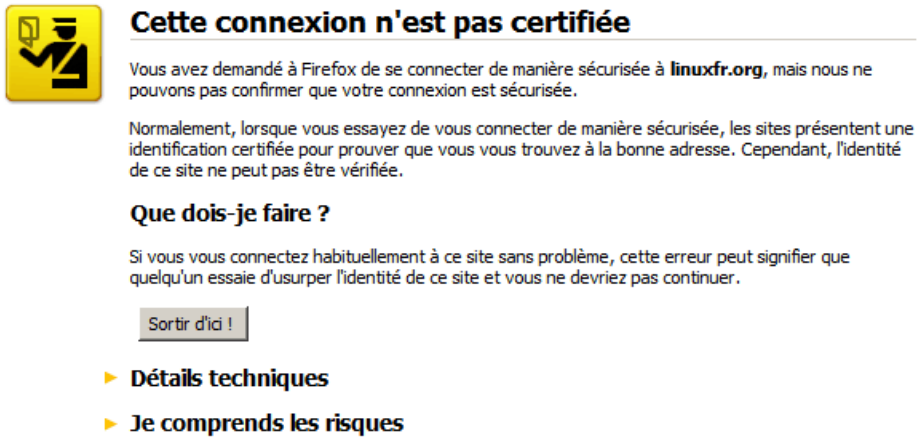
- HTTPsecured : HTTP+SSL
  - Vérification du site grâce à un certificat
  - Utilisation d'un chiffrement asymétrique puis symétrique
  - Certificat de type X.509
    - Signé par une autorité de certification
    - Auto-signé
  - Confidentialité et intégrité des données
  - Utilisation du port 443

  - Utilisation pour toutes les transactions bancaires, financières, achat,…

Https (2)


Gestionnaire de certificats
Vos certificats
Personnes
Serveurs
Autorités
Autres
Vous possédez des certificats enregistrés identifiant ces autorités de certification :
Nom du certificat
Périphérique de sécurité
TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hiz...
TÜRKTRUST Elektronik Sertifika Hizmet Sağlayı...
Builtin Object Token
TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hiz...
TÜRKTRUST Elektronik Sertifika Hizmet Sağlayı...
Builtin Object Token
Unizeto Sp. z o.o.
Unizeto Technologies S.A.
VeriSign, Inc.
Oracle SSL CA - G2
Sécurité personnelle
Voir...
Modifier la confiance...
Importer...
Exporter...
Supprimer ou ne plus faire confiance...
OK


Général
Détails
Ce certificat a été vérifié pour les utilisations suivantes :
Autorité de certification SSL
Émis pour
Nom commun (CN) Oracle SSL CA - G2
Organisation (O) Oracle Corporation
Unité d'organisation (OU) Symantec Trust Network
Numéro de série 31:35:BB:7E:6D:E2:66:3A:B0:E2:33:E6:B0:E1:A5:CC
Émis par
Nom commun (CN) VeriSign Class 3 Public Primary Certification Authority - G5
Organisation (O) VeriSign, Inc.
Unité d'organisation (OU) VeriSign Trust Network
Période de validité
Débute le 06/01/2015
Expire le 06/01/2025
Empreintes numériques
Empreinte numérique SHA-256 E4:AF:2F:AE:41:18:7D:58:F2:09:B0:1B:1D:87:53:C2:DC:CB:3F:60:1C:E8:62:73:E3:7E:87:38:C2:A5:CC:B5
Empreinte numérique SHA1 03:86:D9:39:CF:7B:A7:88:55:27:34:18:5B:EA:D0:B1:3E:87:39:14


Cette connexion n'est pas certifiée
Vous avez demandé à Firefox de se connecter de manière sécurisée à linuxfr.org, mais nous ne pouvons pas confirmer que votre connexion est sécurisée.
Normalement, lorsque vous essayez de vous connecter de manière sécurisée, les sites présentent une identification certifiée pour prouver que vous vous trouvez à la bonne adresse. Cependant, l'identité de ce site ne peut pas être vérifiée.
Que dois-je faire ?
Si vous vous connectez habituellement à ce site sans problème, cette erreur peut signifier que quelqu'un essaie d'usurper l'identité de ce site et vous ne devriez pas continuer.
Sortir d'ici !
Détails techniques
Je comprends les risques

https (3)

- Failles :
  - Man In the Middle
    - Attaque presque invisible
    - Passage de https en http
  - Sécurité interne à l'entreprise
    - On sécurise la communication
    - On ne sécurise pas le stockage des données
    - Est-ce que tous les certificats sont valables ?
    - Pb sur la gestion de la validité…

http/https

- En C
  - Utilisation des sockets habituelles
- En JAVA
  - Utilisation des sockets habituelles
  - Mais deux classes existent pour le http et https:
    - HttpURLConnection

      URL url = new URL(" http://localhost ");
      HttpURLConnection con = (HttpURLConnection) url.openConnection();
    - HttpsURLConnection

      URL url=new URL("https://localhost");
      HttpsURLConnection con = (HttpsURLConnection)url.openConnection();

Gestion des certificats

Défaut de http

- Problèmes :
  - Latence d'une page
    - Latence du réseau , mais disparait avec le haut-débit
    - Augmentation de la rapidité en ne fermant pas la connexion (http 1.1)
    - Utilisation de connexions parallèles pour faire plusieurs requêtes simultanées
      - Mais maximum 8 threads par serveur
    - Utilisation du pipelining dans les requêtes

Amélioration de la latence,
Mise en place de HTTP /2

- basé sur le protocole SPDY
  - Développé par google mais abandonné au profit de http/2

HTTP/2 (1)

- Compatible avec HTTP1.0 et 1.1
  - Même méthode utilisée, même codes de réponse
  - Même URI

- But : être transparent pour l'utilisateur et "plus rapide"

- Tous les serveurs sont compatibles HTTP/2
  - Apache, jetty, microsoft IIS,…

- Les navigateurs web aussi
  - Chrome, firefox, Edge, Safari, Opera…
  - Sous firefox : about:networking

HTTP/2 (2)

- Actuellement , passage de http1.1 en http/2
  - Dans le cas non sécurisé
  - Lors de la 1er requête, rajout d'un champ Upgrade
    - GET / HTTP/1.1 Host: server.example.com
      Connection: Upgrade, HTTP2-Settings
      Upgrade: h2c (http2 clear)
      HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>
  - 2 possibilités :
    - HTTP/1.1 200 OK
      Content-Length: 352
      Content-Type: text/html

      } http2 non supporté
    - HTTP/1.1 101 Switching Protocols
      Connection: Upgrade
      Upgrade: h2c

      } http2 supporté, suite en http2

HTTP/2(3)

- En mode sécurisé
  - Interdiction d'utiliser un upgrade : h2 pour passer de http1.1 à http2s
  - Utilisation d'un protocole au-dessus de TLS
    - Utilisation de l'upgrade h2
    - ALPN : Application-Layer Protocole Negociation
      - Lors de l'échange TLS, prise en compte de l'extension ALPN
      - Inclut dans le java 9
      - Utilisable via openssl

Les navigateurs ont mis en place seulement HTTP/2 pour connexion sécurisée utilisant min TLS1.1

HTTP/2 (4)

- Améliorations
  - **Compression en-tête**
    - Utilisation de la compression
    - Ne pas renvoyer ce qui a déjà été envoyé (cookie, …)
  - **Utilisation de push préventif**
    - Envoie de données qui pourront servir plus tard

Entre 15 à 40 % de gain sur une communication !!!

➢ **HTTP3 : norme en cours de réalisation**

protocole QUIC, sécurisé

grand changement : protocole UDP

Service de messagerie

- Autre nom : courrier électronique, e-mail, couriel
  - ➡ permet d'échanger des messages et des fichiers

- Il nécessite un serveur de messagerie accessible à partir d'internet. Le serveur dispose d'une boîte à lettre (BAL) pour chaque client géré par la messagerie.

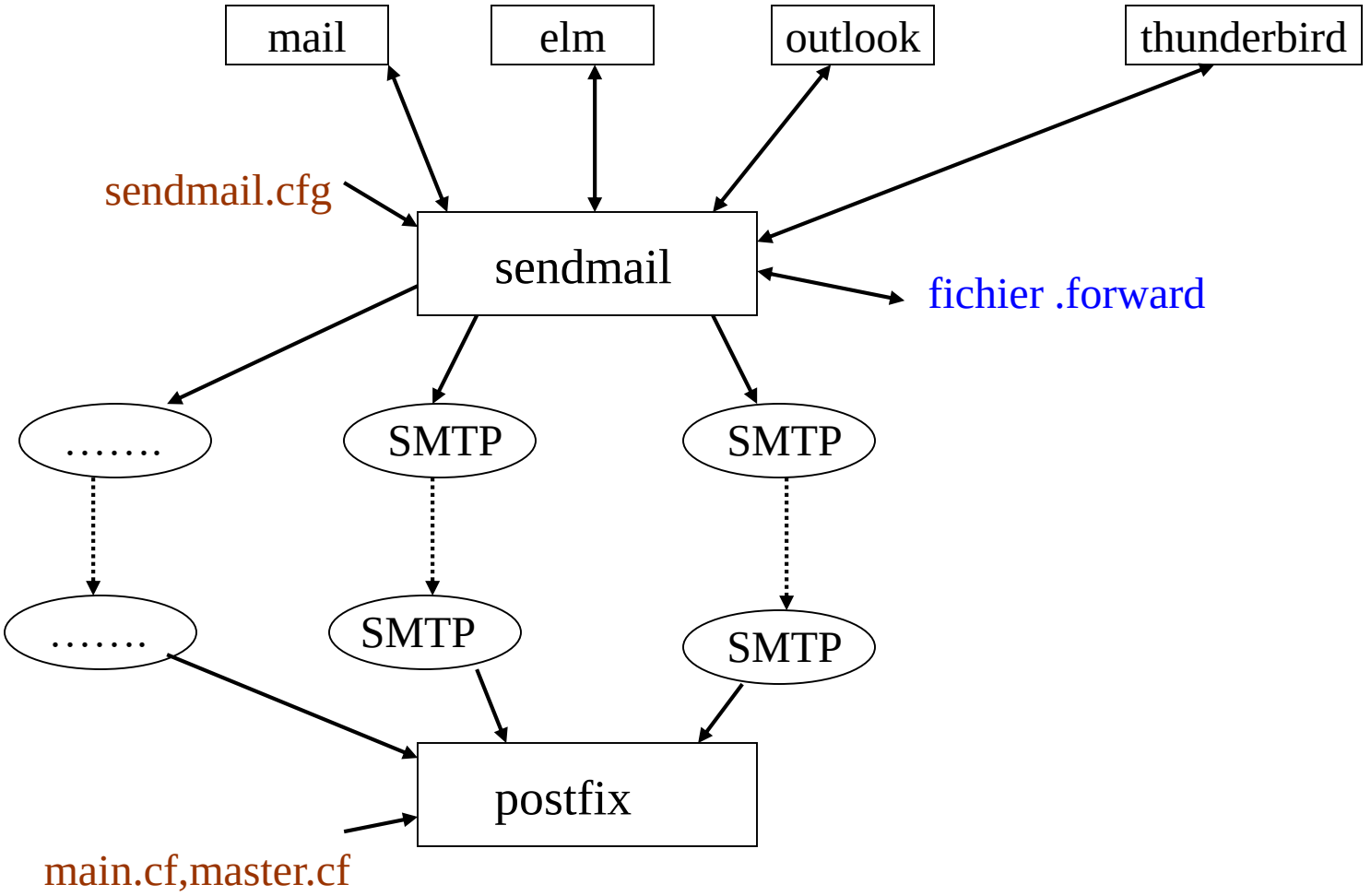
- Les messages sont stockés par le serveur de messagerie, en attendant que le client vienne consulter sa boîte aux lettres.

  - le message peut être lu :
    - en *mode online* - message stocké sur le serveur et lu à distance
    - en *mode offline* – message déplacé sur la station client et effacé du serveur

Terminologie

- Logiciels de messagerie (user agent)
  - Thunderbird, Eudora, Lotus Notes, Outlook, Mail, elm, xmh, …
- Logiciels de gestion de messagerie (Message Transfert Agent)
  - Exchange server, Domino mail server, sendmail, exim, postfix…
  - Hybride : Zimbra
- MSA (Mail Submission Agent)
  - Inclus dans MTA, liaison avec UA
- Protocole de transport des messages (couche applicative)
  - actuellement SMTP,…
- Stockage des messages
  - /usr/spool/mail/…, /var/mail/…

Rôle des entités



SMTP

- SMTP: RFC 821

  Simple Mail Transfert Protocol

  - Permet d'échanger du courrier électronique, protocole orienté texte

- le format des adresses des utilisateurs fait figurer le nom de l'utilisateur suivi du nom de domaine : laurenco@isima.fr

  identifie de manière unique chaque boîte aux lettres (personne@machine.domaine, personne@domaine)

- le codage utilisé pour le message et les fichiers attachés :
  - texte pur codé en ASCII 7 ou 8 bits (RFC 822) pour une prise en compte des caractères accentués
  - standard MIME (Multipurpose Internet Mail Extension) pour du texte formaté, des images ou du son

RFC 822

Structure d'un message

- En tête
- Ligne blanche
- Corps du message (suite de lignes terminés par CR/LF)

En tête

- From : expéditeur
- To : destinataire(s)
- CC : copies à
- Bcc : copie aveugle (destinataire caché)
- Reply-to : adresse de réponse
- Error-to : adresse en cas d'erreur
- Date : date et heure de l'envoi
- Message-id : numéro unique permettant de référencer le message
- Received : informations de transfert
- Subject : sujet

Structure d'un courrier

- Limitations
  - Tout est sous forme de lignes ASCII
  - 2 parties
    - l'entête **(définit les services attendus)**
    - corps **(le texte de la lettre est terminé par une ligne avec "." comme premier et unique caractère)**
  - nom d'utilisateur < 64 caractères
  - nom de domaine < 64 caractères
  - nombre de destinataires < 100
  - une ligne < 1000 catactères
  - utilisation du format MIME (RFC 1521) pour envoyer des fichiers non ASCII

Session SMTP

- Plusieurs phases
  - Connexion TCP sur le port 25
  - Etablissement de la connexion au niveau smtp

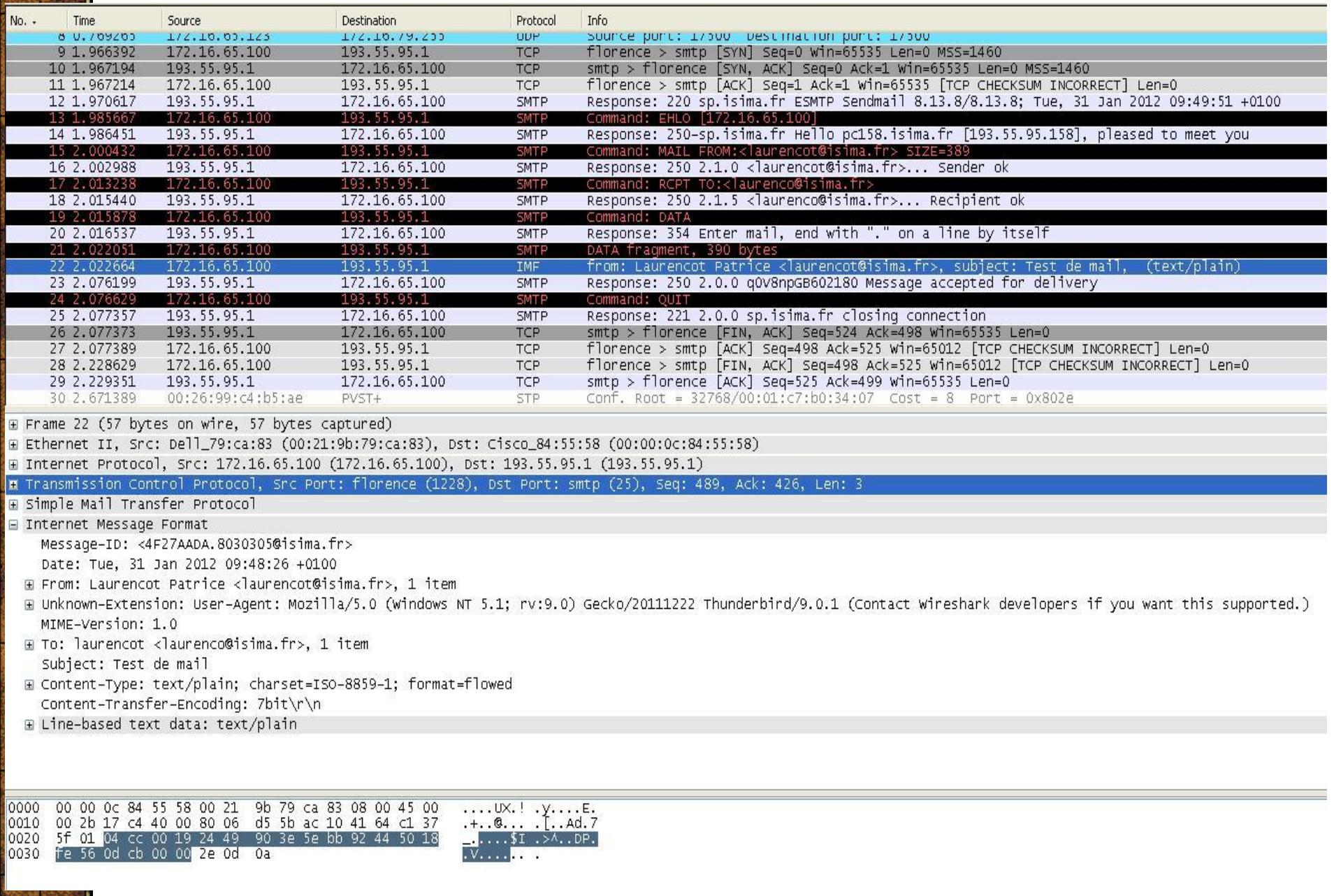
    envoie d'un message HELO et réponse du serveur OK, please to meet you
  - Identification de la source et la destination
  - Envoi du message avec les différents entête
  - Libération de la connexion TCP

- Les "Received" indiquent le chemin suivi, dans l'ordre inverse
  - ils sont ajoutés par les machines (relais SMTP) à travers lesquelles le message a transité
  - ils permettent de retrouver l'origine du message
  - cela évite le bouclage de messages (max 25 champs received)

ESMTP

- Plus récent que SMTP
- Apporte des fonctionnalités supplémentaires
  - taille des messages
  - transport des messages en 8 bits
  - plus de fonctions disponibles

  - Le message de bienvenue est *EHLO.* En cas de réponse négative, le client doit basculer vers l'ancien protocole.


No. Time Source Destination Protocol Info
8 0.769265 172.16.65.123 172.16.79.255 UDP Source port: 17500 Destination port: 17500
9 1.966392 172.16.65.100 193.55.95.1 TCP florence > smtp [SYN] Seq=0 Win=65535 Len=0 MSS=1460
10 1.967194 193.55.95.1 172.16.65.100 TCP smtp > florence [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
11 1.967214 172.16.65.100 193.55.95.1 TCP florence > smtp [ACK] Seq=1 Ack=1 Win=65535 [TCP CHECKSUM INCORRECT] Len=0
12 1.970617 193.55.95.1 172.16.65.100 SMTP Response: 220 sp.isima.fr ESMTP Sendmail 8.13.8/8.13.8; Tue, 31 Jan 2012 09:49:51 +0100
13 1.985667 172.16.65.100 193.55.95.1 SMTP Command: EHLO [172.16.65.100]
14 1.986451 193.55.95.1 172.16.65.100 SMTP Response: 250-sp.isima.fr Hello pc158.isima.fr [193.55.95.158], pleased to meet you
15 2.000432 172.16.65.100 193.55.95.1 SMTP Command: MAIL FROM:<laurencot@isima.fr> SIZE=389
16 2.002988 193.55.95.1 172.16.65.100 SMTP Response: 250 2.1.0 <laurencot@isima.fr>... Sender ok
17 2.013238 172.16.65.100 193.55.95.1 SMTP Command: RCPT TO:<laurenco@isima.fr>
18 2.015440 193.55.95.1 172.16.65.100 SMTP Response: 250 2.1.5 <laurenco@isima.fr>... Recipient ok
19 2.015878 172.16.65.100 193.55.95.1 SMTP Command: DATA
20 2.016537 193.55.95.1 172.16.65.100 SMTP Response: 354 Enter mail, end with "." on a line by itself
21 2.022051 172.16.65.100 193.55.95.1 SMTP DATA fragment, 390 bytes
22 2.022664 172.16.65.100 193.55.95.1 IMF from: Laurencot Patrice <laurencot@isima.fr>, subject: Test de mail, (text/plain)
23 2.076199 193.55.95.1 172.16.65.100 SMTP Response: 250 2.0.0 q0V8npGB602180 Message accepted for delivery
24 2.076629 172.16.65.100 193.55.95.1 SMTP Command: QUIT
25 2.077357 193.55.95.1 172.16.65.100 SMTP Response: 221 2.0.0 sp.isima.fr closing connection
26 2.077373 193.55.95.1 172.16.65.100 TCP smtp > florence [FIN, ACK] Seq=524 Ack=498 Win=65535 Len=0
27 2.077389 172.16.65.100 193.55.95.1 TCP florence > smtp [ACK] Seq=498 Ack=525 Win=65012 [TCP CHECKSUM INCORRECT] Len=0
28 2.228629 172.16.65.100 193.55.95.1 TCP florence > smtp [FIN, ACK] Seq=498 Ack=525 Win=65012 [TCP CHECKSUM INCORRECT] Len=0
29 2.229351 193.55.95.1 172.16.65.100 TCP smtp > florence [ACK] Seq=525 Ack=499 Win=65535 Len=0
30 2.671389 00:26:99:c4:b5:ae PVST+ STP Conf. Root = 32768/00:01:c7:b0:34:07 Cost = 8 Port = 0x802e
Frame 22 (57 bytes on wire, 57 bytes captured)
Ethernet II, Src: Dell_79:ca:83 (00:21:9b:79:ca:83), Dst: Cisco_84:55:58 (00:00:0c:84:55:58)
Internet Protocol, Src: 172.16.65.100 (172.16.65.100), Dst: 193.55.95.1 (193.55.95.1)
Transmission Control Protocol, Src Port: florence (1228), Dst Port: smtp (25), Seq: 489, Ack: 426, Len: 3
Simple Mail Transfer Protocol
Internet Message Format
Message-ID: <4F27AADA.8030305@isima.fr>
Date: Tue, 31 Jan 2012 09:48:26 +0100
From: Laurencot Patrice <laurencot@isima.fr>, 1 item
Unknown-Extension: User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:9.0) Gecko/20111222 Thunderbird/9.0.1 (Contact Wireshark developers if you want this supported.)
MIME-Version: 1.0
To: laurencot <laurenco@isima.fr>, 1 item
Subject: Test de mail
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit\r\n
Line-based text data: text/plain
0000 00 00 0c 84 55 58 00 21 9b 79 ca 83 08 00 45 00 ....UX.! .y....E.
0010 00 2b 17 c4 40 00 80 06 d5 5b ac 10 41 64 c1 37 .+..@... .[..Ad.7
0020 5f 01 04 cc 00 19 24 49 90 3e 5e bb 92 44 50 18 _.....$I .>^..DP.
0030 fe 56 0d cb 00 00 2e 0d 0a .V.....

Format MIME

- Types d'encodage
  - Texte 7 bits, ASCII
  - Texte 8 bits
    - les textes sont composés de caractères 8 bits
    - il faut préciser l'alphabet : ex : iso-latin1 (alias iso-8859-1)
  - Base 64
    - pour les messages binaires
    - groupe de 24 bits, segmentés en 6 bits
      (3 octets, 4*6 bits 0 – 25 → A – Z , 26 – 51 → a –z, 52 – 61 → 0 – 9, 62 → +, 63 → /)
  - Quoted-Printable
    - codage ASCII normal *(A =0x41, a= 0x61, 0=0x30, +=0x2B, /=0x2F)*
    - pour ce qui n'est pas ASCII, valeur = hexa.

Format MIME (3)

- Exemple
  - From :
  - Mime-Version : 1.0
  - To :
  - Subject

  - Content-type : multipart/mixed; boundary="coucou"
    --coucou
    content-type : text/plain; charset=iso-8859-1

    bla-bla
    --coucou
    content-type : audio/basic
    content-transfert-Encoding: base64

    fichier audio
    --coucou

Exemple

POP

- Protocole permettant de relever le courrier sur un serveur
  - POP 3 : Post Office Protocol (version 3) port 110
    - permet l'authentification (login, passwd en clair – codage bin64)
    - réception seulement des courriers sur un serveur (envoi par smtp)
    - réception des messages d'erreur ou d'acquittement

    - Nécessité de télécharger l'intégralité du courrier sur la station avant la lecture, sans possibilité de manipuler directement les messages sur le serveur

    - POP utilise une syntaxe en 4 caractères :
      - STAT : récupère le nombre et la taille des messages en attente
      - LIST ou UIDL : liste des messages à récupérer
      - RETR msg : permet de récupérer un msg
      - DELE msg : suppression
      - USER, PASS : login, passwd
      - QUIT : fin
        (ex : DELE 10, rep = OK, RETR 11, rep=OK+ msg, DELE 11,….)
      - CAPA : Affiche les capacités du serveur

IMAP

- Protocole permettant de relever le courrier sur un serveur
  - IMAP : Internet Message Access Protocol port 143
    - permet l'authentification si nécessaire de façon chiffrée
    - gère les mails sur le serveur, donc pas besoin de télécharger
    - permet de trier les mails, faire des répertoires, etc…
    - utilise des drapeaux pour la gestion des mails

    - IMAP est très utilisé pour les serveurs webmail.

  - Actuellement, version 4 rev 1, RFC 3501
    - Permet de gérer le mode on-line et aussi off-line
    - Utilisation de mots-clés : open, login, list, fetch, logout,…

SPAM

- Courrier non sollicité envoyé à plusieurs personnes
- Actuellement , 95 % mail → SPAM !!
- Les adresses sont récupérées via les listes de diffusion, les pages Web, ou même achetés…

- **Solutions :**
  - Reconnaître l'auteur d'un SPAM
  - Filtrage au niveau personnel
  - Filtrage au niveau d'un site
    - liste noire des "spammeurs" connus
    - refuser les adresse invalides
    - interdire le relayage
    - refuser le mail pour qu'ils soit renvoyé plus tard (efficace, mais cher en BP) -> liste grise