

NOTION DE CRYPTOGRAPHIE



Jean-Louis Gouwy



Plan

- Sécurité informatique
- Moyens utilisés
- Les algorithmes de chiffrement
- Les algorithmes de chiffrement symétriques
- Les algorithmes de chiffrement asymétriques
- Les algorithmes de hachage
- La signature électronique
- Le certificat électronique
 - Quid de authenticité de la clé publique utilisée
 - Mise en œuvre et utilisation
- Algorithme Diffie-Hellman
- Quelques règles
- La sécurité sur Internet
- Webographie et bibliographie



- **Sécurité informatique**

Pour être considéré comme sûr, un échange numérique doit satisfaire les objectifs suivants :

→ **Intégrité**

Les données reçues sont bien celles qui ont été envoyées. Elles n'ont subi aucune transformation durant le transfert.

→ **Confidentialité**

Les données ne pourront être lues en clair (*déchiffrées*) que par la personne à qui elles sont adressées. Un pirate qui les intercepte durant le transfert ne pourra qu'essayer de les *décrypter*.

→ **Authentification**

Consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être.

→ **Non-réputation**

Garantir qu'aucun des correspondants ne pourra nier l'échange de l'information effectuée.



Notion de cryptographie

- **Moyens utilisés**

Des algorithmes de cryptographie s'appuyant sur des modèles mathématiques.

→ **Chiffrement symétrique** (ou chiffrement à clés partagées ou secrètes)

Les données sont chiffrées à l'aide d'un algorithme (fonction) de chiffrement à l'aide d'une clé de chiffrement. La même clé sera utilisée lors du déchiffrement.

→ **Chiffrement asymétrique** (ou chiffrement à trousseau de clés)

Un trousseau de clés (*Clé privée/Clé publique*) sera utilisé par l'algorithme de (dé)chiffrement/déchiffrement.

Les données seront chiffrées à l'aide de la clé publique du destinataire et déchiffrées par celui-ci à l'aide de sa clé privée.

→ **Hachage**

Lorsqu'on applique un algorithme de hachage sur un volume de données (de taille quelconque), on obtient un résultat (*hash ou condensat ou empreinte*) de taille fixe qui identifie ces données de manière unique.



- **Moyens utilisés**

- **Signature électronique**

Est une combinaison de hachage et de chiffrement asymétrique servant à vérifier la provenance (authenticité) et l'intégrité des données.

L'émetteur hachera d'abord ses données. L'empreinte générée sera ensuite signée avec sa clé privée. A la réception, le destinataire vérifiera la signature en déchiffrant celle-ci à l'aide de la clé publique de l'émetteur.

- **Certificat**

Permettra de s'assurer de l'authenticité de la clé publique que l'on utilise. Après tout, n'importe qui peut générer un trousseau de clés et vous envoyer la clé publique en se faisant passer pour quelqu'un d'autre. C'est un fichier associant la clé publique d'une personne (ou d'une machine) à son identité. Le tout étant signé par un tiers de confiance.



Notion de cryptographie

- **Les algorithmes de chiffrement**



Les données sont toujours (dé)chiffrées de la même façon.

Exemples:

- . Chiffrement de César
- . Chiffrement par décalage de bits
- . etc ...

Message en clair (*Plaintext*)

Algorithme de (dé)chiffrement

Cryptogramme (*Ciphertext*)



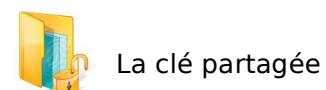
• Les algorithmes de chiffrement symétriques

https://fr.wikipedia.org/wiki/Catégorie:Algorithme_de_cryptographie_symétrique



Les données sont (dé)chiffrées selon un algorithme paramétré par une *clé partagée (clé de chiffrement)*.

La clé partagée est la même au chiffrement et au déchiffrement.



Notion de cryptographie

• Les algorithmes de chiffrement symétriques

Exemples:

- . Chiffrement de César (la clé = la valeur du décalage).
- . Chiffrement par décalage de bits (la clé = la valeur du décalage)
- . Chiffrement par substitution (la clé = le tableau de substitution).
- . Chiffrement de Vigenère

Beaucoup plus robustes

Le décryptage est théoriquement impossible si on ne connaît pas la clé.
Il est souhaitable qu'ils soient connus publiquement afin d'être éprouvés.

- . **DES** (Data Encryption Standard) → Lent et crackable par force brute
- . **3DES** (TripleDES) → Plus difficilement crackable mais très lent
- . **AES** (Advanced Encryption Standard)
 - remplace le DES et le 3DES trop gourmands en ressources, peu performants et assez vulnérables.
- . **RC5** (Rivest Cipher 5) → simple, rapide et facile à mettre en oeuvre.
- . **Blowfish**, etc...

• Les algorithmes de chiffrement symétriques

Avantages

Peu de puissance de calcul → (dé) chiffrement très rapide.

Donc adaptés au chiffrement de grands volumes (transfert réseau, protection des partitions ...)

La **confidentialité** est atteinte. Les données sont chiffrées à l'aide de la clé partagée.

Inconvénients

La clé doit rester secrète:

- Une clé par couple d'intervenants: beaucoup de clés à gérer.
- Une même clé pour tous les intervenants: la communication devient comprise pour l'ensemble des intervenants si seulement un de ceux-ci se fait dérober sa clé.
- Nécessité de communiquer la clé de manière sûre à la personne avec laquelle on souhaite dialoguer.



Notion de cryptographie

• Les algorithmes de chiffrement asymétriques

https://fr.wikipedia.org/wiki/Catégorie:Algorithme_de_cryptographie_asymétrique



Un trousseau de clés est composé d'une clé publique et d'une clé privée.

Un message sera toujours chiffré avec la clé publique du destinataire et ne pourra être déchiffré qu'avec sa clé privée.

Les clés de chiffrement (publiques) et de déchiffrement (privées) sont différentes mais 'complices' ou 'amies'.

La clé publique peut être connue du monde entier mais la clé privée doit être gardée décrete.

Il n'est pas possible de retrouver la clé privée par sa clé publique.



- **Les algorithmes de chiffrement asymétriques**

https://fr.wikipedia.org/wiki/Catégorie:Algorithme_de_cryptographie_asymétrique

Inconvénients

Ces algorithmes sont gourmands en calcul → (dé)chiffrement très lent.

Ils sont donc pas adaptés au chiffrement de grands volumes.

Avantages

Un seul trousseau de clés est nécessaire pour pouvoir communiquer avec x personnes (chaque expéditeur devant simplement connaître la clé publique du destinataire).

La **confidentialité** est atteinte. Les données sont chiffrées à l'aide de la clé publique du destinataire.

L'**authentification** est atteinte. L'expéditeur est certain que seul le détenteur de la clé privée amie pourra déchiffrer le message.



Notion de cryptographie

- **Les algorithmes de chiffrement asymétriques**

Exemples:

- . **RSA:** Rivest Shamir Adleman (1977)

RSA a été breveté par le MIT en 1983 aux États-Unis (brevet expiré 2000). Actuellement, c'est le système à clef publique le plus utilisé (carte bancaire, de nombreux sites web commerciaux...).

- . **DSA:** Digital Signature Algorithm (inventé quand le RSA était encore sous copyright)

Algorithme de signature numérique standardisé par le National Institute of Standards and Technology (NIST) aux États-Unis. Il peut être utilisé gratuitement.

- . **ECDSA :** Elliptic Curve Digital Signature Algorithm (variante de DSA)

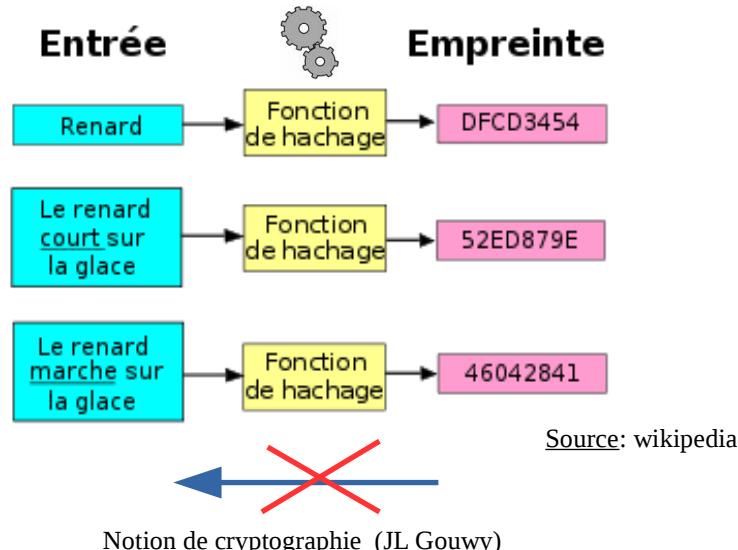
Il fait appel à la cryptographie sur les courbes elliptiques. A sécurité égale, il nécessite une longueur de clé moins longue.

- . etc ...



- **Les algorithmes de hachage**

- . A partir d'une donnée fournie en entrée, cette algorithme calcule une *empreinte* (condensat) servant à identifier rapidement, bien qu'incomplètement, la donnée initiale.
- . Ce type d'algorithme est dit «non-réversible». Il est impossible de retrouver les données initiales à partir de leur empreinte.



Notion de cryptographie (JL Gouwy)

13

Notion de cryptographie

- **Les algorithmes de hachage**

https://fr.wikipedia.org/wiki/Catégorie:Algorithme_de_hachage

- Une empreinte (ou condensat) est calculée et envoyée en même temps que le message sur laquelle elle porte.
- Une empreinte est recalculée sur le message reçu.
- L'**intégrité** est atteinte. On peut considérer que le message n'a pas été modifié durant le transfert si l'empreinte recalculée correspond à celle reçue avec le message.



Notion de cryptographie (JL Gouwy)

14

• Les algorithmes de hachage

Exemples:

- . **MD5** (Message Digest 5): produit un condensat de 128 bits.

Considéré comme obsolète car on a trouvé 2 messages qui génèrent la même empreinte...

Peut accompagner certains téléchargements
(vérification: commande *md5sum*)

- . **SHA** (Secure Hash Algorithm): produit un condensat de 160 à 512 bits

Différentes versions :

SHA-1: considéré comme obsolète
SHA-2: dérivé en plusieurs tailles SHA256, SHA512
SHA-3: alternative à SHA-2

Peut accompagner certains téléchargements
(vérification: commande *shasum* et dérivées)

etc ...

Notion de cryptographie (JL Gouwy)

15



Notion de cryptographie

• Les algorithmes de hachage

Ateliers

a. Utilisation de l'algorithme SHA

- Générez une empreinte sur un fichier.
- Vérification de cette empreinte sur:
 - a) Ce même fichier non modifié.
 - b) Ce même fichier modifié.

b. Vérification de l'intégrité d'un logiciel téléchargé

Vérifiez l'intégrité du package webmin (<http://www.webmin.com>)

1. Téléchargez Webmin (version minimale)
2. Vérification de son empreinte MD5

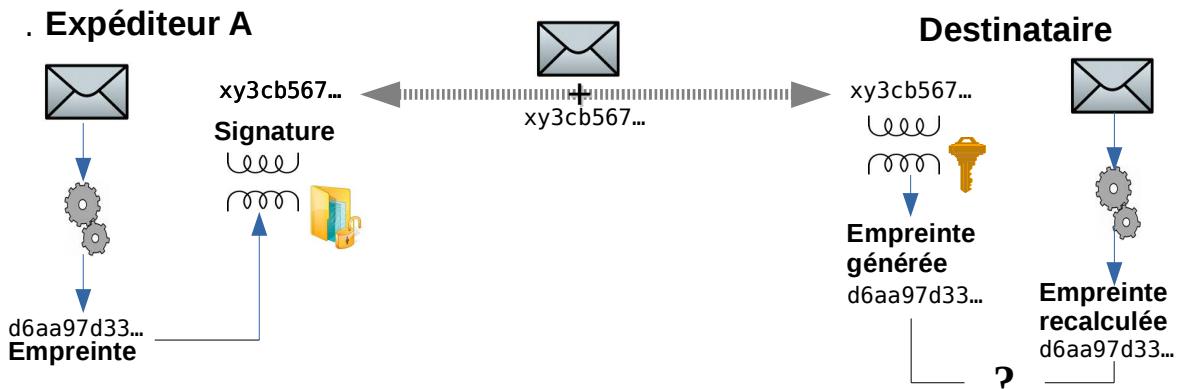


Notion de cryptographie (JL Gouwy)

16

• La signature électronique

- . C'est une combinaison de hachage et de chiffrement asymétrique.



- . A destination, si l'empreinte générée = l'empreinte recalculée

→ L'**intégrité** est atteinte.
 → L'**authenticité** de l'émetteur est atteinte.
 Le destinataire est certain que le message provient bien de l'expéditeur A car son empreinte est signée par celui-ci.



Notion de cryptographie

• Le certificat électronique

- Quid de authenticité de la clé publique utilisée

Comme nous le verrons plus tard,

. SSH permet d'utiliser un simple hash de la clé publique du serveur (fingerprint).
 La vérification de cette empreinte sera à charge du client s'il le désire.
 Cette procédure n'est pas systématique.

. HTTPS, IPSec, OpenVpn... utilisent des **certificats**.

Un **certificat** est un fichier (format x.509 le plus souvent - rfc5280) associant la Clé publique d'une personne (ou d'une machine) à son identité.
 La **non-répudiation** sera atteinte car, dans ce cas, la vérification de l'authenticité de la clé publique sera systématique.



• Le certificat électronique

- Mise en œuvre et utilisation

- a. Création du certificat (CSR : Certificate Signing Request) dont il vaudra faire valider l'identité du propriétaire de la clé publique qu'elle contient.
- b. Envoi du certificat à un tiers de confiance (PKI: Public Key Infrastructure)
- c. Vérification de l'identité du propriétaire du CSR par l'autorité d'enregistrement de la PKI.
- d. Signature du certificat par l'autorité de certification de la PKI et renvoi de celui-ci à son propriétaire.
- e. Vérification de l'authenticité du certificat et de sa clé publique

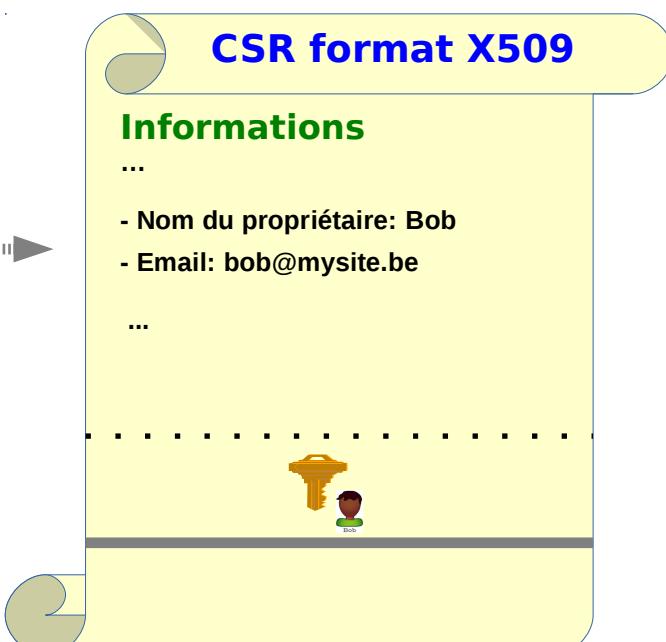
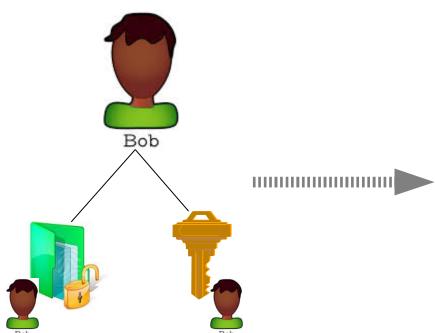


Notion de cryptographie

• Le certificat électronique

- Mise en œuvre et utilisation

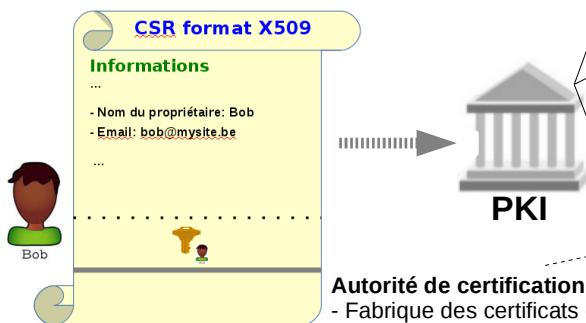
a. Crédit au certificat



• Le certificat électronique

- Mise en œuvre et utilisation

b. Envoi du certificat à un tiers de confiance (PKI)



Autorité de certification (AC)
 - Fabrique des certificats et signe avec sa clé privée.
 - La sécurité de la clé privée de signature est capitale (intégrité).

Ex. de PKI commerciale
 - DigiCert (branches PKI et SSL/TLS de Symantec)
 - ...

Ex. de PKI gratuite
 - Let's Encrypt
 - ...

Une PKI est un ensemble de composantes de technologies de l'information et de dispositifs administratifs structurés dans le but de gérer et réglementer des certificats et des clés publiques/privées associés à des entités physiques.

Les services d'une PKI

- Enregistrement d'utilisateurs.
- Gestion des certificats (création, distribution, renouvellement et révocation) et des clés associées.
- Archivages des certificats (sécurité et recouvrement en cas de perte)

Structure d'une PKI

Autorité d'enregistrement (AE)
 - Indispensable, fait le lien avec la personne physique et l'AC.
 - Gère les demandes et vérifie leurs validités.

Service de publication
 - Donne accès aux certificats (annuaire).

Service de révocation
 - Gestion d'une liste des certificats expirés ou invalides.

Notion de cryptographie

• Le certificat électronique

- Mise en œuvre et utilisation

c. Vérification de l'identité du propriétaire par la PKI

L'autorité d'enregistrement (AE) demande une information que seule la personne légitime possède :

Exemples

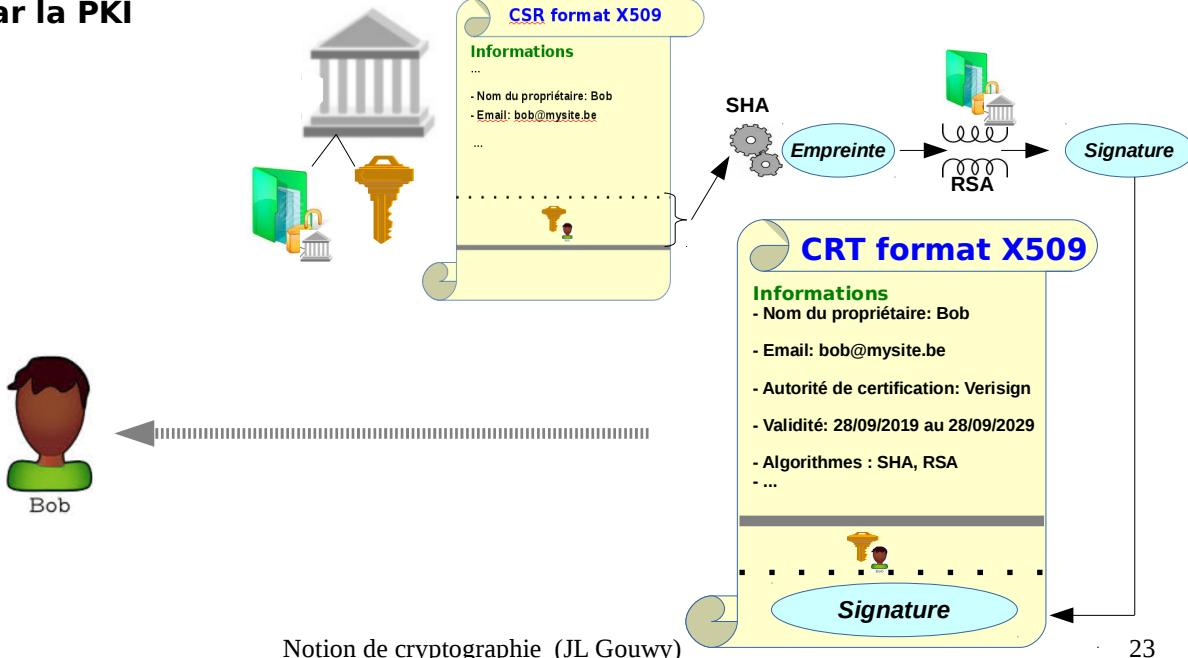
Pour un domaine, envoi d'un mail au gestionnaire du domaine avec un lien à cliquer pour être sûr que c'est bien lui qui est à l'origine de la demande du certificat.

Pour une validation étendue (Extended Validation – EV), l'AE ira jusqu'à demander une photocopie de la carte d'identité ou du passeport du demandeur.

• Le certificat électronique

- Mise en œuvre et utilisation

d. Signature du certificat par la PKI



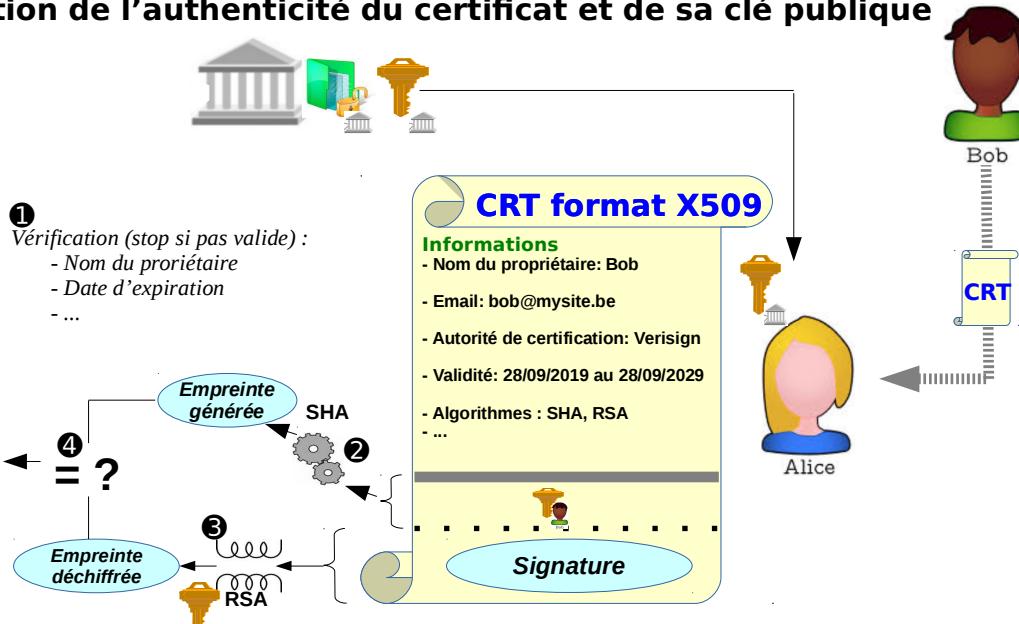
Notion de cryptographie

• Le certificat électronique

- Mise en œuvre et utilisation

e. Vérification de l'authenticité du certificat et de sa clé publique

Si les 2 empreintes sont identiques, cela garantit à Alice que le certificat (contenant la Kpub) de Bob a bien été validé par la PKI. Ainsi, la provenance du certificat (et de sa clé publique) est validée.



• Algorithme Diffie-Hellman

C'est une méthode par laquelle deux personnes nommées conventionnellement Alice et Bob peuvent se mettre d'accord sur un nombre (qu'ils peuvent utiliser comme clé pour chiffrer la conversation suivante) sans qu'une troisième personne appelée Eve puisse découvrir le nombre en écoutant.

Source: wikipedia

Cet algorithme permet d'établir entre le client et le serveur une clé de chiffrement symétrique (*clé de session*) en échangeant seulement des valeurs publiques.

Remarque

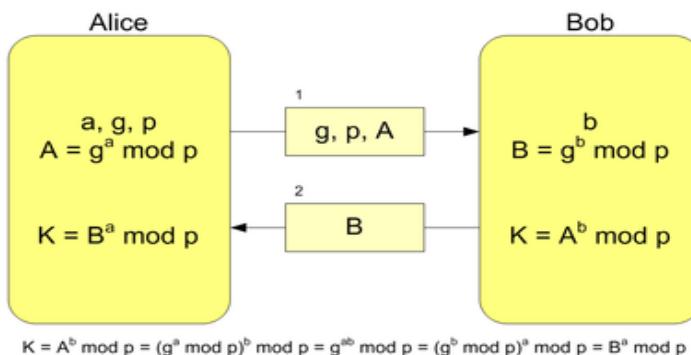
Les clefs secrètes sont aussi appelées clefs de session, clefs symétriques ou clés partagées.



Notion de cryptographie

• Algorithme Diffie-Hellman

Principe



- Alice choisit un nombre aléatoire **a** et Bob choisit un nombre aléatoire **b**.
- Alice et Bob s'entendent sur 2 autres nombres **g** et **p**
(ils peuvent circuler en clair sur le réseau car il sera difficile d'inverser les exponentiations suivantes).
- Alice calcule $A = g^a \text{ mod } p$ et Bob calcule $B = g^b \text{ mod } p$.
- A est envoyé à Bob et B est envoyé à Alice.
- Alice calcule $B^a \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = (g^{ba} \text{ mod } p) = K$
- Bob calcule $A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = (g^{ab} \text{ mod } p) = K$

Source: wikipedia

K = la clé secrète



• Algorithme Diffie-Hellman

Exemple	CLIENT	SERVEUR
Chaque partie génère un nombre aléatoire.	a=2	b=3
Les parties se mettent d'accord sur 2 nombres communs.	g=3 p=7 – <i>Echange en clair</i> –	g=3 p=7
Chaque partie calcule $g^{nbalea} \text{ mod } p$	$\begin{aligned} g^a \text{ mod } p \\ = 3^2 \text{ mod } 7 \\ = 2 \end{aligned}$	$\begin{aligned} g^b \text{ mod } p \\ = 3^3 \text{ mod } 7 \\ = 6 \end{aligned}$
Echange du résultat	6 – <i>Echange en clair</i> –	2
Chaque partie calcule la clé de session : NbReçu ^{nbalea} mod p	$\begin{aligned} 6^2 \text{ mod } 7 \\ = 1 \end{aligned}$	$\begin{aligned} 2^3 \text{ mod } 7 \\ = 1 \end{aligned}$
	$\begin{aligned} \downarrow \\ = 6^a \text{ mod } p \\ = (g^b \text{ mod } p)^a \text{ mod } p \\ = g^{ba} \text{ mod } p \end{aligned}$	$\begin{aligned} \downarrow \\ = 2^b \text{ mod } p \\ = (g^a \text{ mod } p)^b \text{ mod } p \\ = g^{ab} \text{ mod } p \end{aligned}$
	=	=



Notion de cryptographie

• Quelques règles

- . La **vitesse de transfert** est d'autant plus rapide que la clé de session est courte; mais le degré de confidentialité est d'autant plus élevé que la clé de session est longue.
- . L'expéditeur **chiffre** avec la clé publique du destinataire et le destinataire déchiffre avec sa clé privée correspondante.
ex. Si tous mes contacts chiffraient le contenu de leur mail avec ma clé publique avant de me l'envoyer, ils seraient assurer que moi seul pourrait le lire (le déchiffrer)
→ **confidentialité**
- . L'expéditeur **signe** avec sa clé privée et le destinataire vérifie la signature avec la clé publique correspondante.
ex. Si, à l'aide de ma clé privée, je générerais une signature avant l'envoi d'un mail à un de mes contacts, celui-ci serait assurer que ce mail provient bien de moi (après vérification de la signature avec ma clé publique)
→ **authentification**
- . On peut **chiffrer et signer** un échange réseau.
ex. Un de mes contacts m'envoie un mail chiffré (avec ma clé publique) et signé (avec sa clé privée). Mon client mail dispose de ma clé privée (pour lire le message) et de sa clé publique (pour vérifier l'identité de mon contact)
→ **confidentialité et authentification**



- **La sécurité sur Internet**

- . **PGP** (*Pretty Good Privacy*)

Les PKI ne sont plus utilisés. Les certificats sont signés par une chaîne d'ami(s)...

Usages:

Messagerie électronique

Divers: chiffrement de données, de disques...



Notion de cryptographie

- **La sécurité sur Internet**

- . **SSL/TLS** (*Secure Socket Layer/Transport Layer Security*)

Ensemble de protocoles^(*) permettant de sécuriser un paquet au niveau de la couche transport.

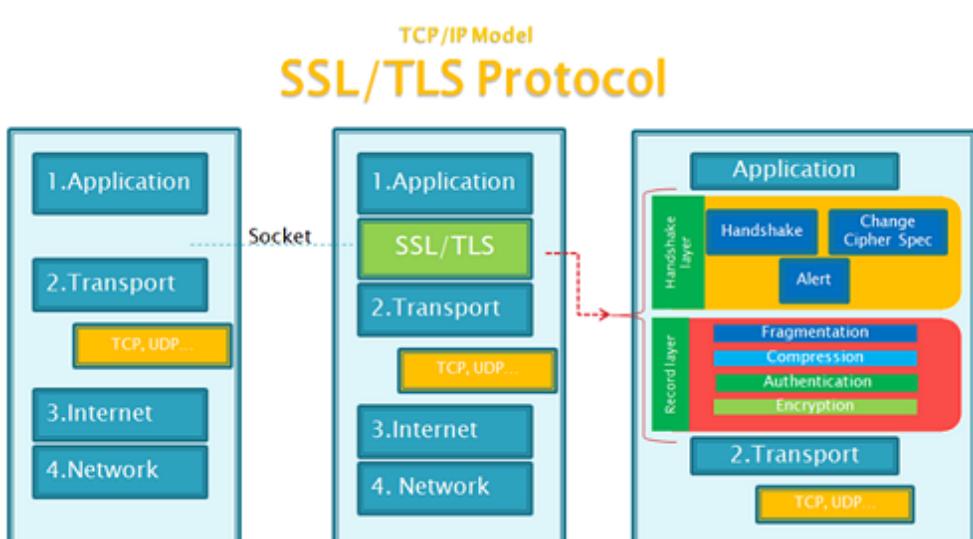
(*)

Handshake protocol
Etablissement de la session

ChangeCipherSpec protocol
Négociation des algorithmes

Alert protocol
Messages d'erreurs

Record protocol
Protection des données



- **La sécurité sur Internet**

- . **SSL/TLS**

Usages: HTTPS, SSH, FTPS, IMAPS, ... mais aussi OpenVPN pour les VPN

Critiques:

- . Ne nécessite pas l'emploi de protocoles spécifiques de niveau application.
- . Relative lenteur car les données doivent traverser une couche intermédiaire.
- . Ne permet pas de protéger les données de bas niveau: ICMP, IGMP, RIP...
- . Pas de problème si les paquets sont natés ...



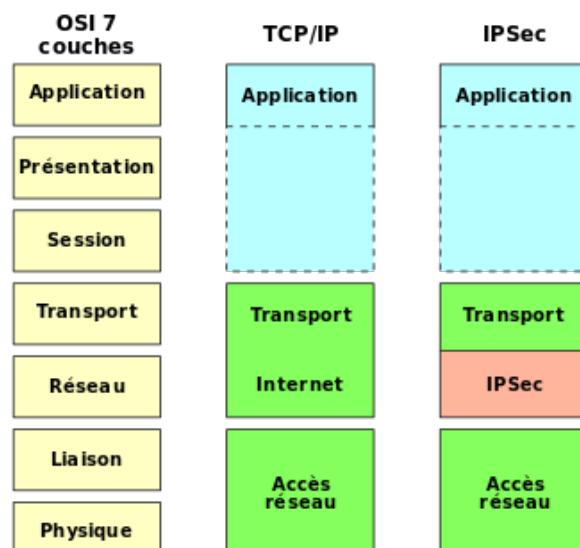
Notion de cryptographie

- **La sécurité sur Internet**

- . **IPsec (IPSecure)**

Ensemble de protocoles^(*) permettant de sécuriser un paquet au niveau de la couche réseau (OSI) ou de la couche IP (TCP/IP).

- (*) **IKE (Internet Key Exchange)**
Négociation de la connexion
- AH (Authentification Header)**
Authentification des paquets en les signant → le paquet ne peut pas être modifié durant le transfert (intégrité) !!!!
- ESP (Encapsulating Security Payload)**
Chiffrement des données



- **La sécurité sur Internet**

- . **IPsec**

- 2 modes de fonctionnement

- Le mode 'transport': ce sont uniquement les données transférées qui sont chiffrées et/ou authentifiées.
 - Le mode 'tunnel': c'est la totalité du paquet IP qui est chiffré et/ou authentifié. Le paquet est ensuite encapsulé dans un nouveau paquet IP avec un nouvel en-tête IP.

Usages: VPN très souvent.

Critiques:

- . Plus rapide car les données ne doivent pas traverser une couche intermédiaire.
- . L'implémentation n'est pas toujours aisée en IPv4 mais devient native dans IPv6.
- . Permet de protéger les données de bas niveau: ICMP, IGMP, RIP...
- . Problème si, en mode 'transport', les paquets doivent traverser un routeur Nat...



Notion de cryptographie

- **La sécurité sur Internet**

- . **SSH (Secure Shell)**

. C'est un protocole permettant de:

- se connecter à distance
 - d'échanger des fichiers

et ce, de manière entièrement sécurisée.

. L'ensemble de la session est chiffrée à l'aide de nombreux algorithmes de la bibliothèque SSL/TLS.

- . **VPN (Virtual Private Network)**

- . Il permet de connecter 2 sites distants à travers un réseau non sécurisé (comme l'Internet). Une fois la connexion réalisée, le flux de données entre les 2 sites sera entièrement sécurisé.
 - . La connexion et le flux de données transitant entre les 2 sites pourra être orchestrée par SSL/TLS ou IPsec.



• La sécurité sur Internet

. Firewall (Coupe-feux)

Dispositif installé entre le réseau interne d'une entreprise et l'Internet.
Leur but est d'autoriser (ou empêcher) certains paquets de poursuivre leur route.

2 types de firewalls

- Les firewalls dit "filtrant": Ils agissent au niveau de la couche réseau (Ip) ou transport.
- Les firewalls dit "proxy": Ils agissent au niveau de la couche applicative.



Références

WEBOGRAPHIE

- <http://www.commentcamarche.net/contents/crypto/crypto.php3>
- http://www.formation.jussieu.fr/irt/2007-2008/SECURITE/cours/Crypto_Appliquee.pdf
- http://www.authsecu.com/cours-formation-elearning/cours-formation-elearning.phphttps://access.redhat.com/documentation/fr-fr/red_hat_enterprise_linux/7/pdf/system_administrators_guide/Red_Hat_Enterprise_Linux-7-System_Administrators_Guide-fr-FR.pdf
- <https://www.youtube.com/watch?v=UMKCYIJloKo>
- <https://www.slideserve.com/marilu/introduction-la-cryptographie>
- <https://www.frameip.com/ssl-tls/>
- <https://www.frameip.com/ipsec/>

+ wikipedia...

BIBLIOGRAPHIE

INITIATION A LA CRYPTOGRAPHIE
Par Gilles Dubertert - Edition Vuibert Informatique

RED HAT ENTREPRISE LINUX/CENTOS
Mise en production et administration de serveurs
Par Thibault Bartolone - 2e éd.
2015 - Editions ENI (2e éd.)



Laboratoire de sécurité internet

SECURE SHELL



Jean-Louis Gouwy



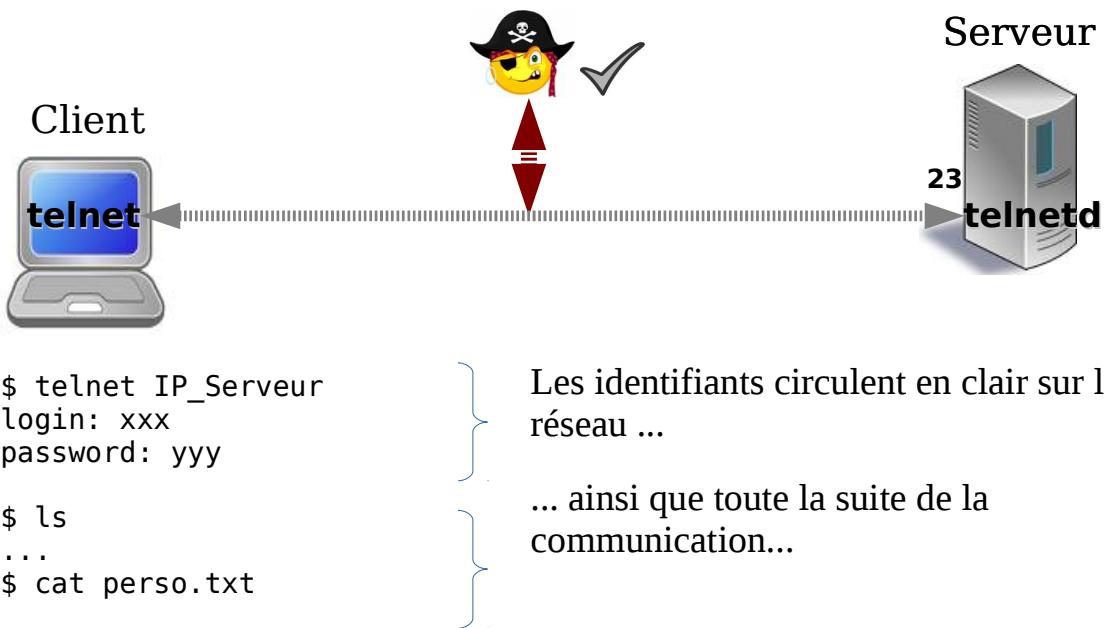
Plan

- Introduction
- Qu'est-ce que OpenSSH ?
 - Utilités et avantages
- Etablissement d'une communication
- Configuration du serveur
- Implémentation par mot de passe
 - Remarques
 - Exercice
- Implémentation par clés
 - Remarques
 - Exercice
- Avantages fonctionnels et techniques
- Gestion d'un parc Linux
 - Implémentation de la solution
 - Exercices
- Quelques outils
- Le port forwarding
 - Exercice
- Le tunneling
- Références



Introduction

- **Une conversation non-sécurisée: telnet**



Introduction

- **Atelier: Espionner une conversation telnet**

- Lancer *Wireshark* sur le poste client.
- Définir un filtre de capture pour telnet (*tcp.port==23*) et pour l'interface concernée.
- Lancer la capture.
- Etablir une connexion telnet.
- Analyser le contenu des trames en vue d'y retrouver les identifiants.



- OpenSSH repose sur une bibliothèque de fonctions (OpenSSL) qui contient tous les éléments cryptographiques nécessaires au fonctionnement d'OpenSSH.
- C'est la version OpenSource (développée par OpenBSD sous licence BSD) de la suite logicielle proposée par la société SSH Communication Security.
- C'est une application (client/serveur) permettant d'accéder à une machine distante de façon sécurisée.
- C'est aussi le nom du protocole réseau associé à cette application
 - ssh version 1 : obsolète
 - ssh version 2 : actuellement utilisé
- Il remplace notamment les applicatifs *telnet* et *rsh* qui sont considérés comme non-sécurisés.



Utilités et avantages

- Obtenir un accès distant en garantissant trois des quatre objectifs légaux à atteindre lors de l'échange de données :
 1. Authentification: Seules les personnes autorisées ont accès aux ressources.
 2. Intégrité: On est sûr que le message n'a pas été modifié.
 3. Confidentialité: On est sûr que le message n'a pas pu être lu en clair.

Rem: La 'non répudiation' qui garantit qu'une transaction ne peut être niée par aucune des 2 parties n'est pas assurée par ssh car l'obtention de la clé publique n'est pas toujours sûre... (voir plus loin).

```
# ssh -l pgouwyjl natrezo.bacisat.be ou ssh pgouwyjl@natrezo.bacisat.be
```

- Exécuter une commande à distance de façon sécurisée.
`# ssh pgouwyjl@natrezo.bacisat.be cat /etc/shadow`
- Faire du transfert de fichiers sécurisé (scp, secpanel, winscp ou sftp)
- Encapsuler un autre protocole (tunneling)
- Faire du X11 forwarding (prendre en main le bureau graphique d'une machine distante par ssh → encapsulation du protocole X dans un tunnel ssh).



Etablissement d'une communication

Step 1: Négociation entre le client et le serveur

	CLIENT	SERVEUR
S1 Négociation	<ul style="list-style-type: none"> . Se mettent d'accord sur les crypto-systèmes à utiliser par la suite et la version du protocole ssh à utiliser (ex. ssh2) <ul style="list-style-type: none"> Ex. RSA pour le chiffrement asymétrique AES pour le chiffrement symétrique SHA pour le hachage . K_{pubs} $\xleftarrow{\hspace{1cm}}$ K_{pubs}/K_{privs} 	

Remarque

Lors de l'acceptation du téléchargement automatique de la clé publique (étape 1), le client accepte, peut-être à tort, que la machine la proposant soit le bon serveur...

- ▶ Transmettre 'manuellement' la clé publique... ou vérifier le *fingerprint* (voir plus loin)



Openssh (JL Gouwy)

7

Etablissement d'une communication

Step 2: Authentification du serveur

	CLIENT	SERVEUR
S2 Authentification du serveur	<ul style="list-style-type: none"> (Le serveur doit répondre à un défi) 	$7+5 \rightarrow RSA \xleftarrow{\hspace{1cm}} K_{pubs} \xrightarrow{\hspace{1cm}} RSA \rightarrow 7+5$ $K_{privs}, \sqrt{ }$
✓ Authentification	✓ $12 \xleftarrow{\hspace{1cm}} 12$	ASYMETRIQUE

Step 3: Etablissement d'un canal sécurisé

S3 Ouverture d'un canal sécurisé	<ul style="list-style-type: none"> . Echange sécurisé d'une clé de session (Diffie-Hellman) $K_{ses} \xleftarrow{\hspace{1cm}} K_{ses}$. Les data qui seront échangées par la suite passeront par un canal ssh $\rightarrow AES \xleftarrow{\hspace{1cm}} K_{ses} \xrightarrow{\hspace{1cm}} AES \rightarrow K_{ses}$
----------------------------------	--

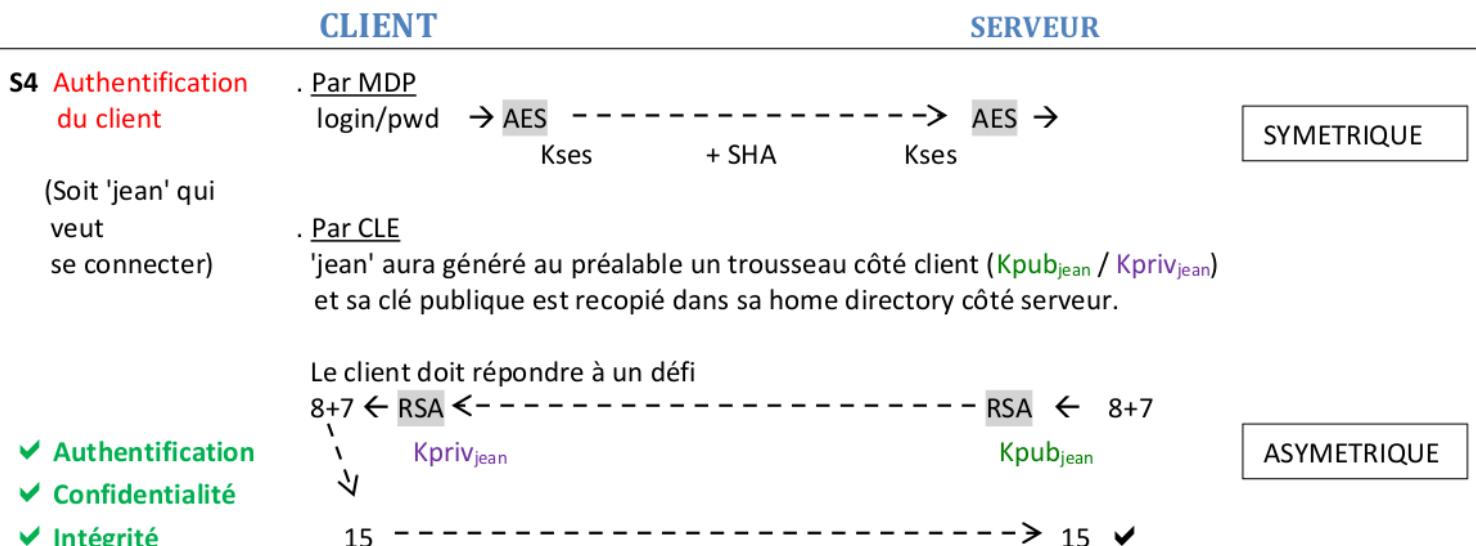


Openssh (JL Gouwy)

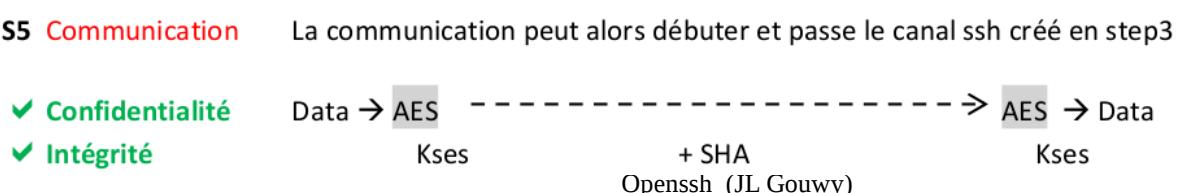
8

Etablissement d'une communication

Step 4: Authentification du client



Step 5: Authentification du client



Etablissement d'une communication

Avantages d'une authentification par clés

- Connexion et communication sécurisée sans mot de passe.
- Décorellation des mots de passe SSH et Linux.
Pas besoin de reconfigurer le mot de passe Linux si on se fait dérober sa clé privée ...)
- Le login de connexion dépend de l'endroit où sera déposée la clé publique du client. Ainsi, déposer ma clé publique dans la home directory de l'utilisateur 'root' de plusieurs serveurs me permettrait d'être 'root' sur ces serveurs sans entrer de mot de passe.
- Très utile lors de sauvegardes sécurisées, autonomes et automatisées par scripts.

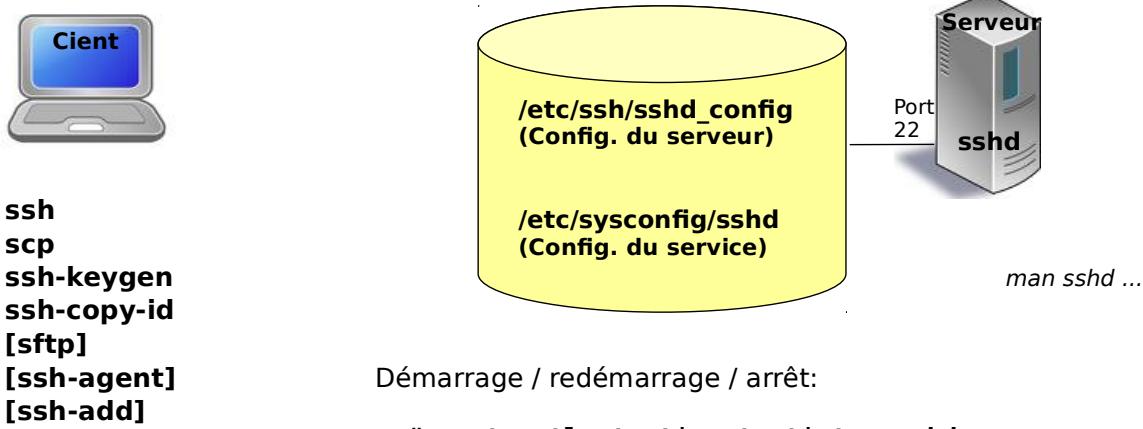


- **Atelier: Espionner une conversation ssh**

- Lancer *Wireshark* sur le poste client.
- Définir un filtre de capture pour ssh (*tcp.port==22*) et pour l'interface concernée.
- Lancer la capture.
- Etablir une connexion ssh.
- Constatations : Négociation des algorithmes, Diffie-Hellman, chiffrement des paquets ...



Configuration du serveur



Les versions d'OpenSSH ≥ à 7.4 sont implémentées pour travailler uniquement via le protocole SSHv2.



Configuration du serveur

Le fichier sshd_config: exemple

Les versions d'OpenSSH ≥ à 7.4 sont implémentées pour travailler uniquement via le protocole SSHv2.

Port 22	→	Port d'écoute de sshd
AddressFamily any	→	Quel type d'adresse sshd utilise (inet, inet6 ou any)
ListenAddress 0.0.0.0	→	@Ip de l'interface d'écoute (ici toutes les interfaces ipv4)
ListenAddress ::	→	@Ip de l'interface d'écoute (ici toutes les interfaces ipv6)

HostKey /etc/ssh/ssh_host_rsa_key	{	<i>Localisation de toutes les clés privées du serveur</i> → de type RSA → de type DSA → Variante DSA (basée sur les courbes elliptiques) → Implémentation de l'EdDSA <i>(Variante DSA basée sur les courbes elliptiques tordues d'Edwards)</i>
HostKey /etc/ssh/ssh_host_dsa_key		
HostKey /etc/ssh/ssh_host_ecdsa_key		
HostKey /etc/ssh/ssh_host_ed25519_key		
...		

LoginGraceTime 2m → Temps accordé à la procédure de login

OpenSSH (JL Gouwy)

13



Configuration du serveur

PubkeyAuthentication yes → Authentification par paire de clés SSH2 acceptée.

HostbasedAuthentication no
IgnoreRhosts yes } → Pour empêcher les authentifications de type remote (car non-sécurisées).

PasswordAuthentication yes → Authentification par mot de passe (rabattement possible en cas d'échec à l'authentification par clés).

PermitEmptyPasswords no → Mais pas pour les comptes sans mdp.

TCPKeepAlive yes	{	<i>Pour éviter que la connexion reste ouverte si le client disparaît, le serveur coupe la connexion s'il ne reçoit plus du client un message « Je suis en vie » envoyé régulièrement par celui-ci.</i>
ClientAliveInterval 15		
ClientAliveCountMax 10		

OpenSSH (JL Gouwy)

14



Configuration du serveur

Le fichier sshd_config: Autres directives

DenyUsers
AllowUsers test admin

Autoriser les deux utilisateurs (test et admin) et aucun autre à se connecter.

Voir aussi les directives
AllowGroups/DenyGroups

PermitRootLogin yes → Le root peut-il se connecter ?

PermitRootLogin without-password

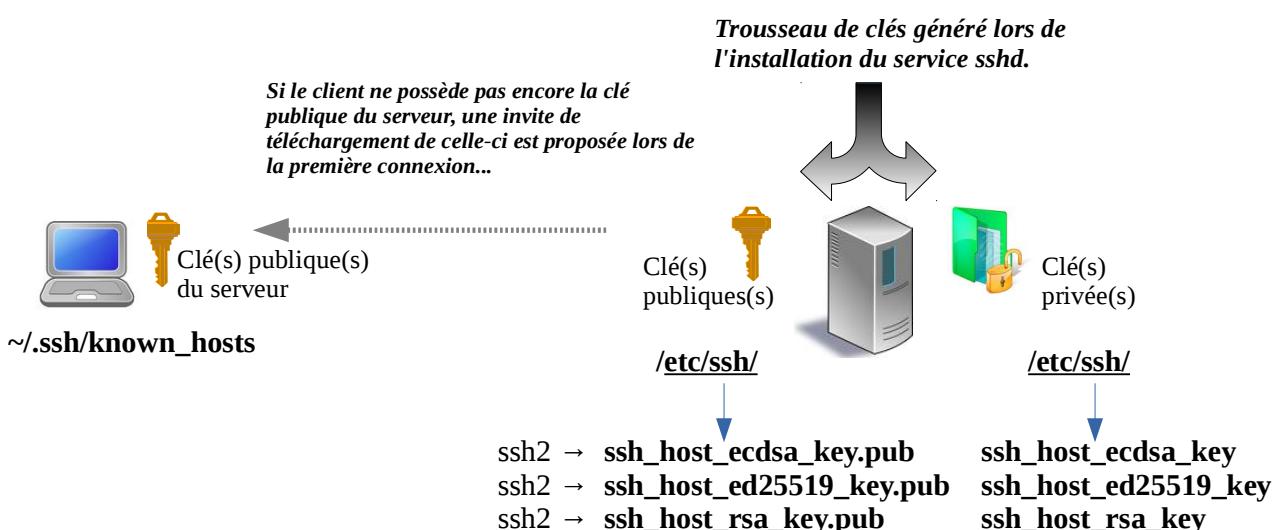
Le root ne peut se connecter que par paire de clés. Cela évite les tentatives d'attaque ssh par force brute sur le compte root.



Openssh (JL Gouwy)

15

Implémentation par mot de passe



Rem.

ssh-keygen -t rsa -b 1024 → pour générer une clé de 1024 bits ...

ssh-keygen -R 192.168.1.100 → pour supprimer une clé du fichier known_hosts



Openssh (JL Gouwy)

16

Implémentation par mot de passe

Le fingerprint

Lors du téléchargement de la clé publique, le client devrait s'assurer qu'il s'agit bien de celle du serveur ciblé au risque de se connecter à un pirate qui aurait usurpé l'IP du vrai serveur.

Une bonne méthode:

- L'administrateur du serveur génère un 'fingerprint' (chaîne générée lors de la génération de la clé publique du serveur).

La commande 'ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key.pub' permet d'afficher cette empreinte.

```
256 SHA256:I66kqJq80+JIDdICHImXQW6NULn/kz8wfXHezzB3mHQ no comment (ECDSA)
```

- Le client le demande à l'administrateur et le compare à celui présenté lors de la demande de téléchargement de la clé publique.

```
The authenticity of host '192.168.1.1 (192.168.1.1)' can't be established.
ECDSA key fingerprint is SHA256:I66kqJq80+JIDdICHImXQW6NULn/kz8wfXHezzB3mHQ.
ECDSA key fingerprint is MD5:c3:49:e5:f9:94:05:94:53:aa:ae:9a:16:4a:a6:74:55.
Are you sure you want to continue connecting (yes/no)?
```

- ✓ Cette procédure ne garantit en rien la non-réputation car elle n'est pas obligatoire et systématique.

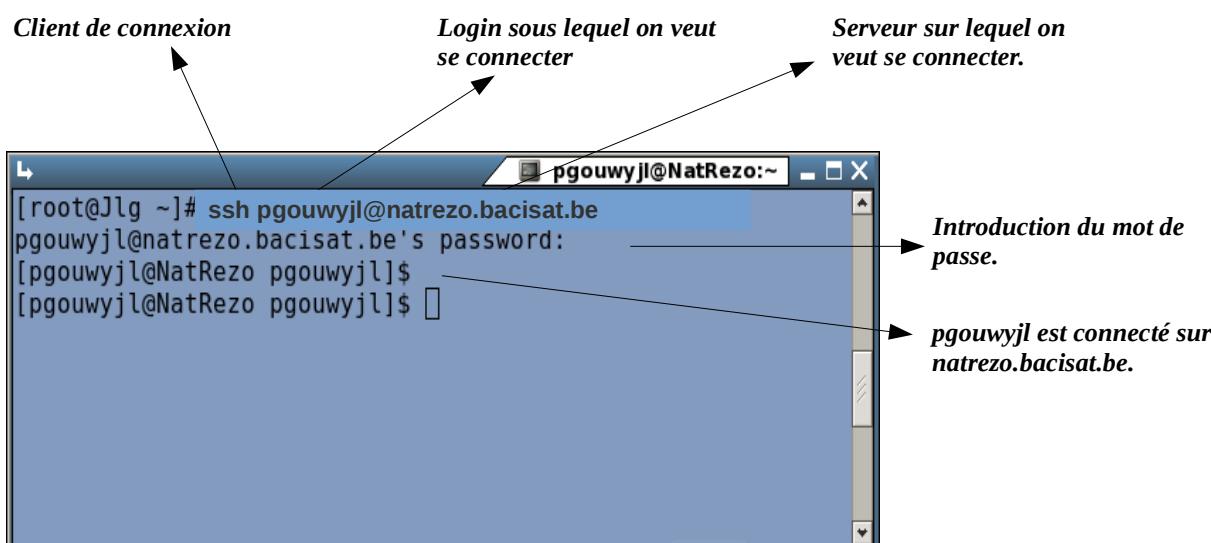
Openssh (JL Gouwy)

17



Implémentation par mot de passe

Exemple

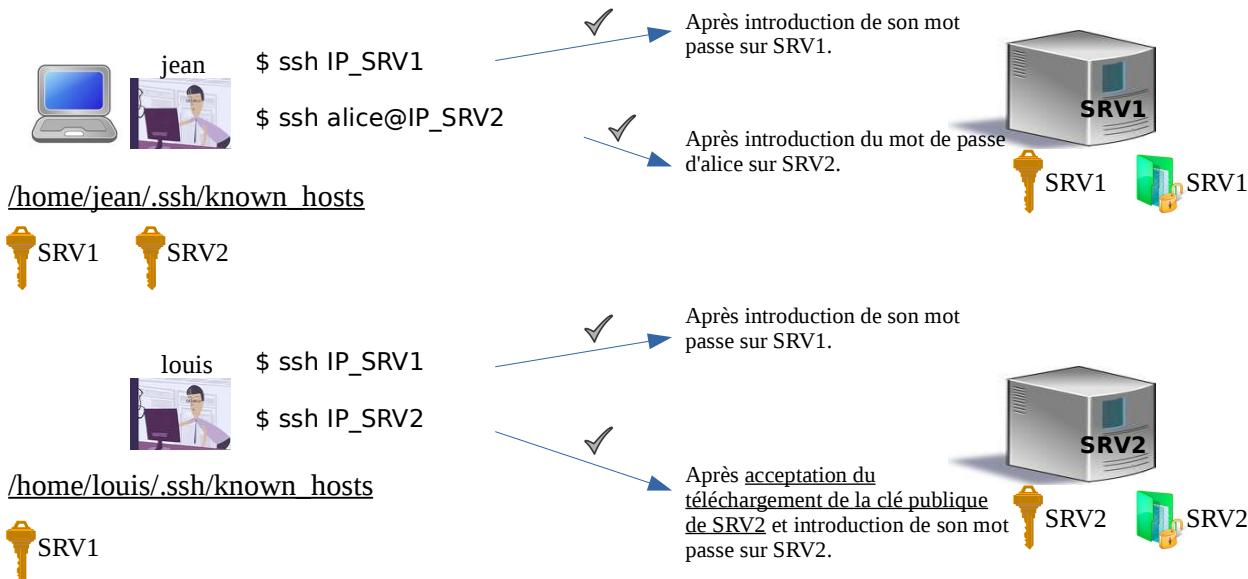


Openssh (JL Gouwy)

18

Implémentation par mot de passe

On peut trouver un fichier **known_hosts** dans plusieurs home directories.
 Il peut contenir plusieurs clés publiques appartenant à des serveurs différents.

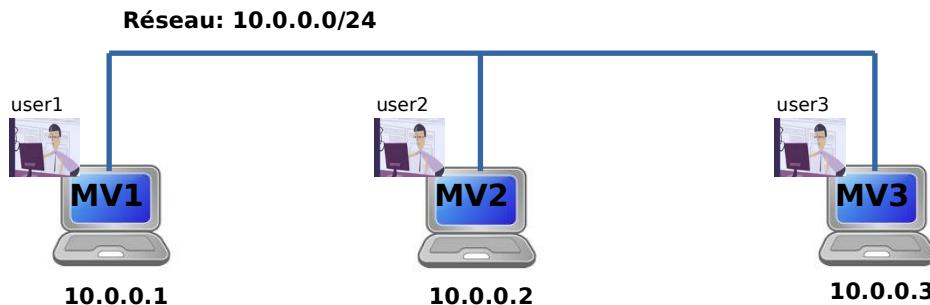


Remarques

- Attention aux permissions sur les fichiers et répertoires (voir labo.)
- Fichiers logs: `/var/log/messages` & `/var/log/secure`
- `/usr/sbin/sshd -d` pour lancer le serveur en mode debug...
- Le protocole SSH1 souffre de failles de sécurité dans l'intégrité des données envoyées ou reçues, le rendant vulnérable à des attaques actives.
- Mieux vaut configurer le serveur pour n'accepter que les connexions sous protocole SSH2.
- SFTP n'est opérationnel que sous le protocole SSH2.



Exercice: Connexion par mot de passe



Chaque machine doit accepter des connexions par mot de passe.

Pour ce faire:

- vérifiez sur chaque machine qu'un trousseau de clés existe
- configurez le service sshd sur chaque machine et lancez le service (bien vérifiez que la directive PasswordAuthentication est à yes)
- créez user1 sur MV1, user2 sur MV2 et user3 sur MV3



Exercice: Connexion par mot de passe

- testez les connexions suivantes:

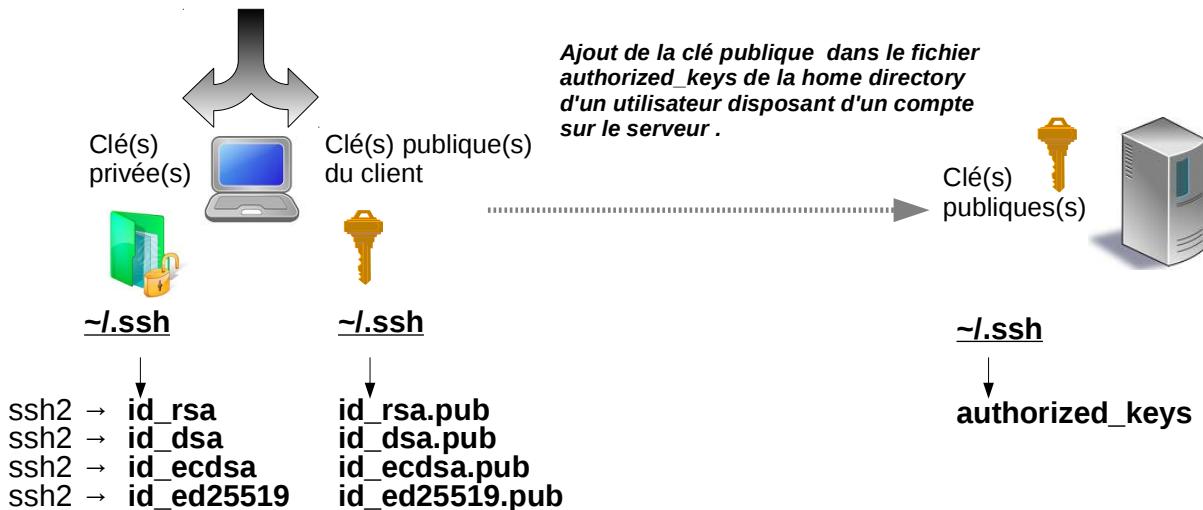
user1@MV1\$ ssh user2@10.0.0.2	user2@MV2\$ ssh user1@10.0.0.1
user1@MV1\$ ssh user3@10.0.0.3	user2@MV2\$ ssh user3@10.0.0.3
user3@MV3\$ ssh user2@10.0.0.2	
user3@MV3\$ ssh user1@10.0.0.1	

- visualisez les ports d'écoute (client/serveur) à travers une connexion
- utilisez ssh et scp (après avoir consulter les pages de manuel)
- explorez les fichiers /etc/ssh/* et ~/.ssh/known_hosts



Implémentation par clés

```
# ssh-keygen -t rsa
# ssh-keygen -t dsa
# ssh-keygen -t ecdsa
# ssh-keygen -t ed25519
```



Openssh (JL Gouwy)

23

Implémentation par clés

Exemple



Openssh (JL Gouwy)

24

Implémentation par clés

On peut trouver un fichier **authorized_keys** dans plusieurs home directories.
 Il peut contenir plusieurs clés publiques appartenant à des clients différents.

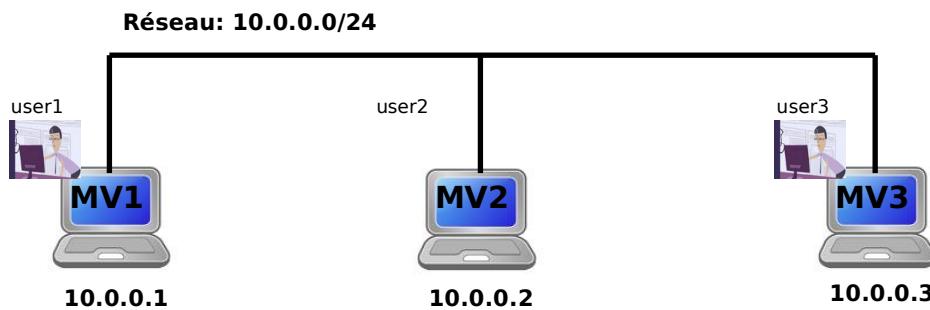


Remarques

- Si une autre personne a accès à votre compte (ou vole votre disque dur), il pourrait copier votre clé privée et s'en servir pour accéder à votre compte sur un serveur SSH sauf si elle a été protégée par une PASSPHRASE (cryptée via l'algorithme des) lors de sa génération ...
- L'agent ssh: Outil qui charge en mémoire la clé privée pour éviter d'avoir à rentrer la PASSPHRASE plusieurs fois ...



Exercice : Connexion par clés



Faites en sorte que *user1* puisse se connecter sur MV3 sous *user3* sans mot de passe.

Pour ce faire:

- root@MV3# → vérifie la clause PubkeyAuthentication à yes dans /etc/ssh/sshd_config
- user1@MV1\$ → génère un trousseau de clés de type RSA (sans passphrase)
- user3@MV3\$ → crée un dossier ~/.ssh
- user1@MV1\$ → copie sa clé publique dans un fichier authorized_keys de user3 (attention aux permissions)
- user1@MV1\$ → se connecte sous user3 sur MV3.

Explorez les fichiers ~/.ssh/* et ~/.ssh/authorized_keys



Utilités et avantages

Avantages fonctionnels du Secure Shell:

Pouvoir travailler de manière sécurisée sur une machine sans être obligé de se déplacer.

Pouvoir facilement transférer des fichiers de manière sécurisée.

Pouvoir facilement centraliser et automatiser l'administration d'un ensemble de machines (scripts).



Utilités et avantages

Avantages techniques du Secure Shell:

Authentification forte:

- si connexion par paire de clés (pas de circulation de mot de passe)
- protection supplémentaire: possibilité de protéger sa clé privée par une passphrase.
- algorithme de chiffrement asymétrique très puissant (RSA/DSA/ECDSA/ED25519)
- possibilité de vérifier la validité de la clé publique du serveur lors de son téléchargement via son empreinte (fingerprint)

Alerte contre les attaques de type 'man-in-the-middle':

- si lors d'une connexion, le message REMOTE HOST IDENTIFICATION HAS CHANGED apparaît chez le client, cela signifie que la clé publique du serveur n'est plus la même que celle reçue lors de la 1ère connexion...

- l'administrateur a regénéré un nouveau trousseau de clés
- ou
- un pirate tente de détourner votre connexion



Gestion d'un parc Linux

- Mise à jour logicielle.
- Changement de configuration.
- Supervision des stations.
- ...

Bien souvent, les mêmes manipulations à répéter sur des dizaines de stations 😞.

☞ Besoin d'automatiser ces tâches.

Une solution: ssh



Gestion d'un parc Linux

Solution 1:

Intervenir localement et effectuer les interventions à la main sur chaque station.

 Nécessité de se déplacer.

Risque d'erreur.

Tâche trop lourde si le parc est important et décentralisé.

Solution 2:

Se connecter via ssh (par mot de passe) sur chaque station et effectuer les interventions à distance et manuellement sur chaque station.

 Risque d'erreur.

Tâche trop lourde si le parc est important.
(1000 stations = 1000 connexions + 1000 interventions)



Gestion d'un parc Linux

Solution 3:

Se connecter via ssh (par mot de passe) sur chaque station et automatiser à distance les interventions (scripts) sur chaque station.

 Tâche trop lourde si le parc est important.
(1000 stations = 1000 connexions)

Solution 4:

Se connecter via ssh (par clé) sur chaque station et automatiser à distance les interventions (scripts) sur chaque station.

SOLUTION RETENUE



Implémentation de la solution

Sur la station d'administration

- Un trousseau de clés doit être généré.
- Un fichier /root/admin/ip.txt contiendra les @ip de toutes les stations à administrer.
- Un dossier /root/admin/logs contiendra les logs générés par les scripts d'administration.
- Un dossier /root/admin/sh contiendra les scripts d'administration.

Sur les stations du parc

- La clé publique de root de la station d'administration doit être recopiée dans la home directory de root de chaque station (fichier authorized_keys).
- sshd doit tourner.



Exercices: Gestion d'un parc Linux

Exercice 1:

Ecrivez et testez un script (pushkey.sh) qui déploie la clé publique de root de la machine d'administration (MV1) vers toutes les stations du parc.

Exercice 2:

Ecrivez et testez un script (haltall.sh) qui éteint toutes les stations du parc encore « on-line ».

Programmez l'exécution de ce script à 21h00 tous les jours.
Pour ce faire le package crontab doit être installé...



Exercices: Gestion d'un parc Linux

Exercice 3:

Ecrivez et testez un script (`chpwdroot.sh`) qui change mot de passe de root sur toutes les stations du parc. Le nouveau de passe est passé en argument au script.

Exercice 4:

Ecrivez et testez un script (`alladduser.sh`) qui ajoute un compte utilisateur à chaque station du parc.

Le nom de l'utilisateur (ex. `toto`) sera passé en argument. Son mot de passe sera identique à son nom.



Exercices: Gestion d'un parc Linux

Exercice 5:

Ecrivez et testez un script (`chgrub.sh`) qui permet de changer le 'timeout' du multi-boot de chaque machine à 5 secondes.

Exercice 6:

Ecrivez et testez un script (`chnetwork.sh`) qui change la configuration ip de chaque station du parc afin de les disposer sur le réseau d'adresse `192.168.0.0/24`.

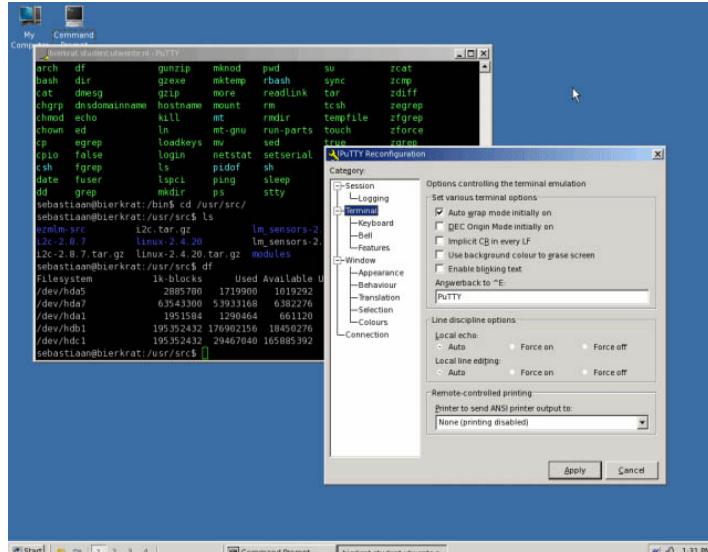


Quelques outils

Pour Windows et Linux

PUTTY: Utilitaires mode texte pour clients telnet et ssh , copie de fichiers de manière sécurisée, générateur de trousseau de clés...

☛ <http://www.putty.org/>



Openssh (JL Gouwy)

37

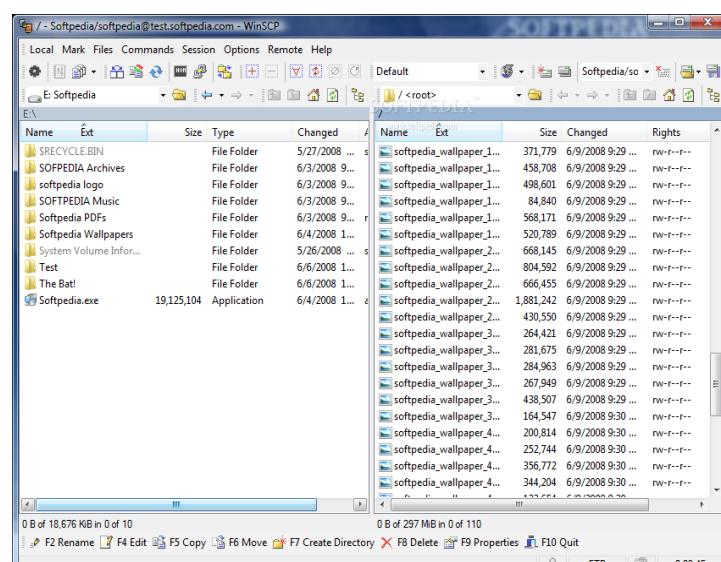


Quelques outils

Pour Windows

WINSCP: Utilitaire graphique permettant le transfert de fichiers par 'drag and drop'.

☛ <http://winscp.net/eng/docs/lang:fr>



Openssh (JL Gouwy)

38

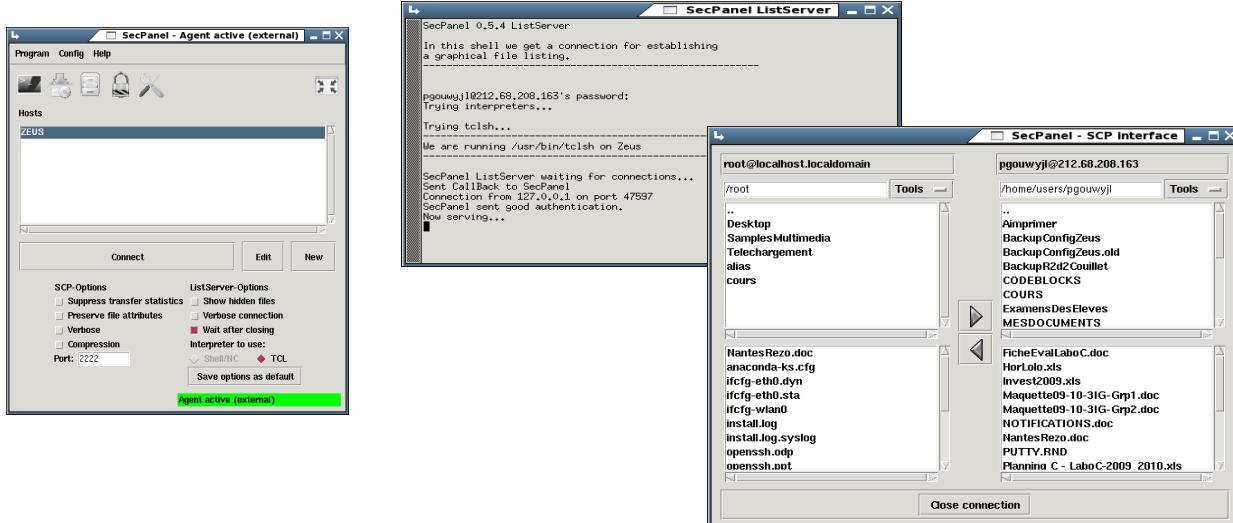


Quelques outils

Pour Linux

secpanel: Utilitaire graphique (écrit en Tcl/Tk) permettant de combiner les fonctionnalités de PUTTY et de WINSCP.

► <http://themediahost.de/secpanel/>



Openssh (JL Gouwy)

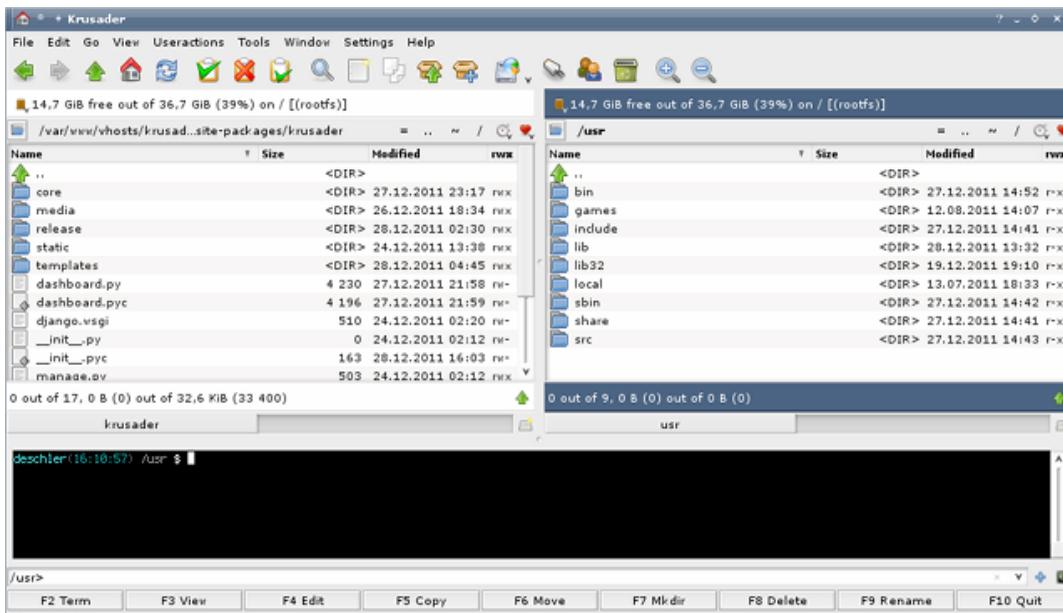
39

Quelques outils

Pour Linux

Krusader: Utilitaire graphique permettant le transfert de fichiers par 'drag and drop'.

► <http://www.krusader.org>



Openssh (JL Gouwy)

40



Quelques outils

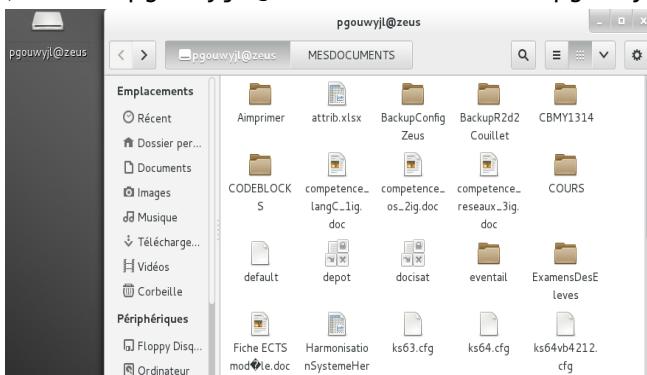
Pour Linux

sshfs: Sert à monter un fs distant, à travers une connexion SSH, le tout avec des droits utilisateur. L'avantage est de manipuler les données distantes avec n'importe quel gestionnaire de fichier (Nautilus, Konqueror ou même la ligne de commande), ce qui est bien plus pratique que la commande scp couplée avec ssh.

Bonne alternative à certains utilisateurs Windows qui utilisaient WinSCP.

- ☛ http://www.server-world.info/en/note?os=CentOS_6&p=fuse
- ☛ <http://doc.ubuntu-fr.org/sshfs>

```
$ sshfs pgouwyjl@zeus.moi.be:/home/pgouwyjl /home/my_dir
```



Openssh (JL Gouwy)

41

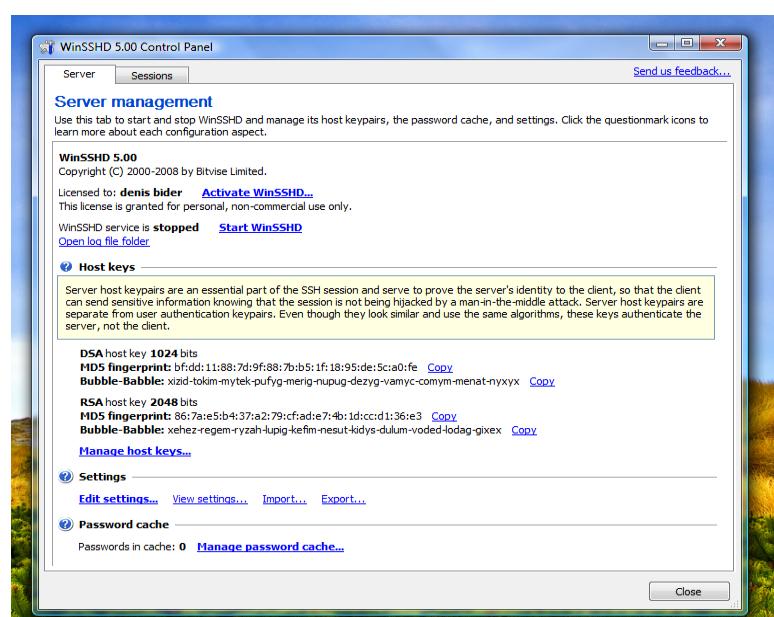
Quelques outils

Pour Windows

Windows 10: Serveur SSH en natif.

WinSSHD: Serveur SSH pour Windows.

- ☛ <http://www.bitvise.com/winsshd>

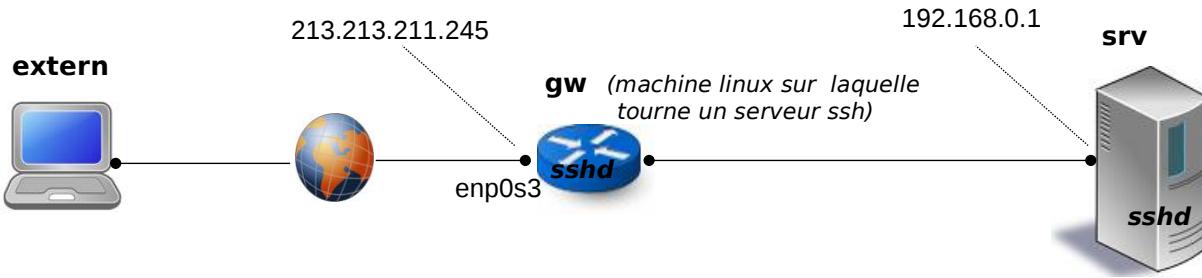


Openssh (JL Gouwy)

42

Le port forwarding

- Maquette



- Comment se connecter sur **srv** à partir de **extern** ?

1° **extern\$** ssh 213.213.211.245

gw\$ ssh 192.168.0.1

srv\$

} ☹ 2 connexions et sshd nécessaire sur gw



Le port forwarding

2° Faire du port forwarding

Sur gw, détourner les **connexions tcp entrantes par enp0s3 sur un port spécifique (ex. 2222)** vers une autre machine (srv) sur son port ssh (22).

```
gw# iptables -t nat -I PREROUTING -p tcp -i enp0s3 --dport 2222 -j DNAT --to 192.168.0.1:22
```

```
gw# iptables -L -t nat    (Pour vérifier)
```

```
extern# ssh -p 2222 213.213.211.245  
(Introduire le mot de passe)
```

```
srvlou$ who  
root pts/2 <ip_Ad_extern>
```

☺ 1 connexion et sshd nécessaire uniquement sur srv



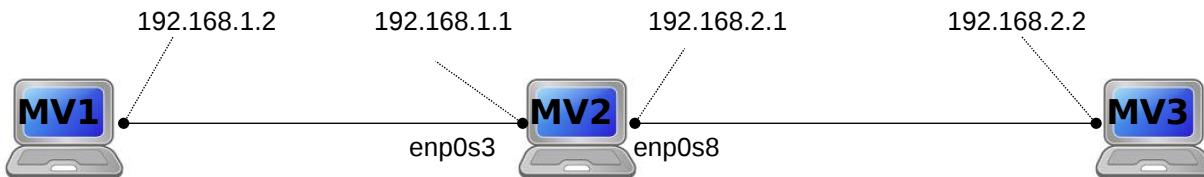
- **Remarques**

- Le port cible (ici 2222) doit être disponible sur la passerelle.
- Commande iptables: voir chapitre 'Le firewalling et Netfilter'
- Le forwarding de port est configurable sur n'importe quel routeur physique actuel.
- N'importe quel port peut être forwardé vers n'importe quel autre port.
(ex. Le port 8080 de gw vers le port 80 d'une autre machine ou même le port 80 de gw vers le port 80 d'une autre machine ...)
- Avantage: permet d'accéder à des services qui tournent derrière une passerelle sur un réseau privé et protégé par firewall, proxy ...



Exercice: Le port forwarding

- **Maquette**



- Réalisez la maquette présentée ci-dessus (virtualisation: réseau interne)
- Configurez et testez un forwarding de port de MV1 vers MV3 en passant par MV2

Remarque

- Le port forwarding implique l'ip forwarding.
- MV1 et MV3 ne peuvent pas se toucher mutuellement (le routage n'est pas configuré).
- Le nating sur MV2 doit être activé. Pourquoi ?



- SSH permet de véhiculer n'importe quel flux TCP dans un tunnel chiffré d'une session SSH. C'est le petit cousin du VPN (Virtual Private Network)
- Donc, le flux de l'application considérée (très souvent non chiffré) sera encapsulé à l'intérieur du "tunnel" SSH. Il ne sera donc plus véhiculé en clair entre les ports client et serveur habituels.
- Le tunneling repose sur 2 grands principes: la redirection (ou relayage) de ports et l'encapsulation de protocoles.
- Applications classiques possibles:
 - Tunneliser sa lecture de mails sur un client POP3.
 - Tunneliser toute une connexion et un flux Ftp.
 - Tunneliser une conversation http.
 - Tunneliser un accès à une base de données.
 - ...
- Si votre firewall (ou proxy) bloque certaines applications mais pas le port 22, celles-ci pourront néanmoins être disponibles à condition d'être véhiculées par un tunnel ssh...



Le tunneling

Le tunneling peut s'opérer de 2 façons différentes:

- Local (**-L**)

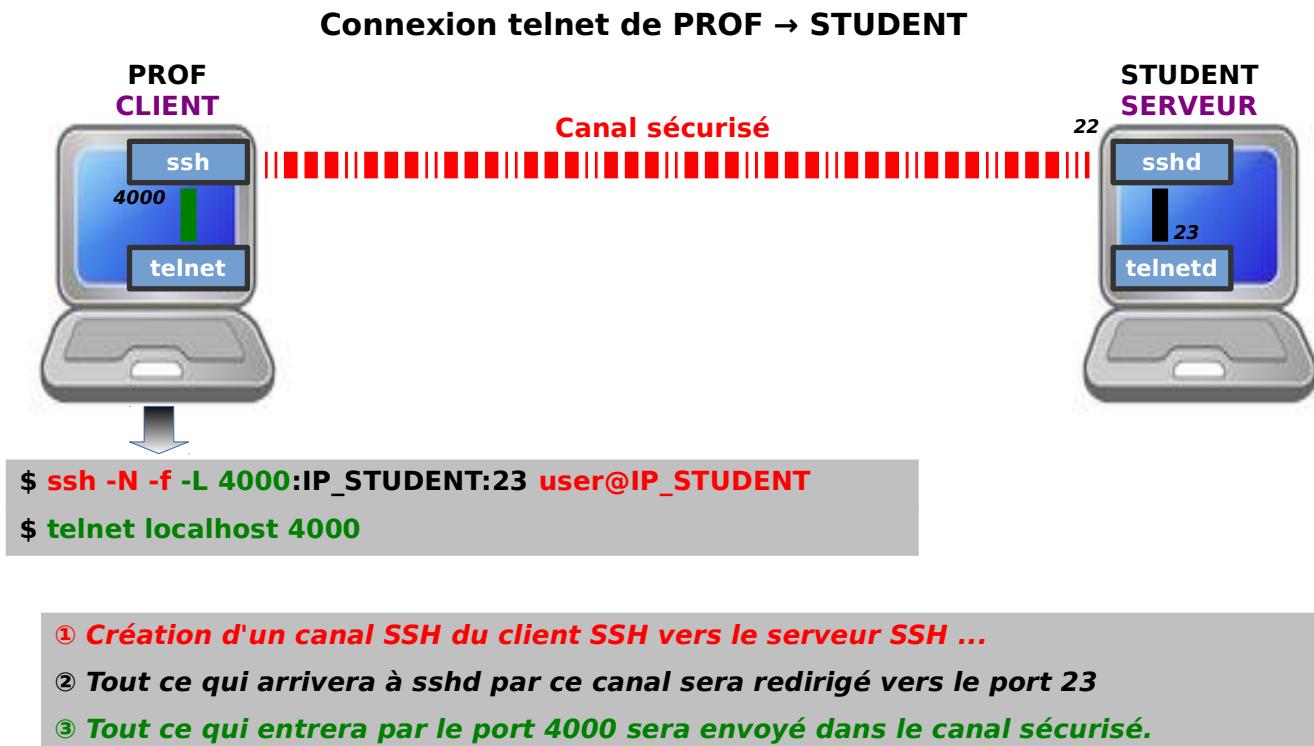
C'est le cLient qui devra créer le tunnel vers le service désiré puis l'utiliser.

- Remote (**-R**)

C'est le seRveur qui devra créer le tunnel et mettre à la disposition du client un service.



Local - Modèle à 2 machines - Soit 'tunneliser' une conversation telnet...



Openssh (JL Gouwy)

49

Le tunneling

Local - Modèle à 2 machines - Constatations

- L'option -N permet de n'exécuter aucune commande distante.
- L'option -f permet de lancer ssh en background.
- Le port local (ici 4000) doit être libre sur la machine concernée.
- Tout ce qui transite par le canal est chiffré donc sécurisé.
Seules les données transitant en local dans les 2 machines ne sont pas chiffrées.
- Pour utiliser le canal, il suffit d'envoyer ses données sur le port 4000 de la machine locale (`telnet localhost 4000`).
- Un modèle à 2 machines implique l'utilisation des 2 mêmes adresses ip dans la commande.
- Le canal reste ouvert même après fermeture de la communication telnet.
- Le client n'a besoin d'aucun daemon serveur sur sa machine mais la création du tunnel et son utilisation est à sa charge. De plus, il a besoin d'un compte sur le serveur.
- Le serveur sshd pourrait être le point de convergence de plusieurs canaux.



Openssh (JL Gouwy)

50

Le tunneling

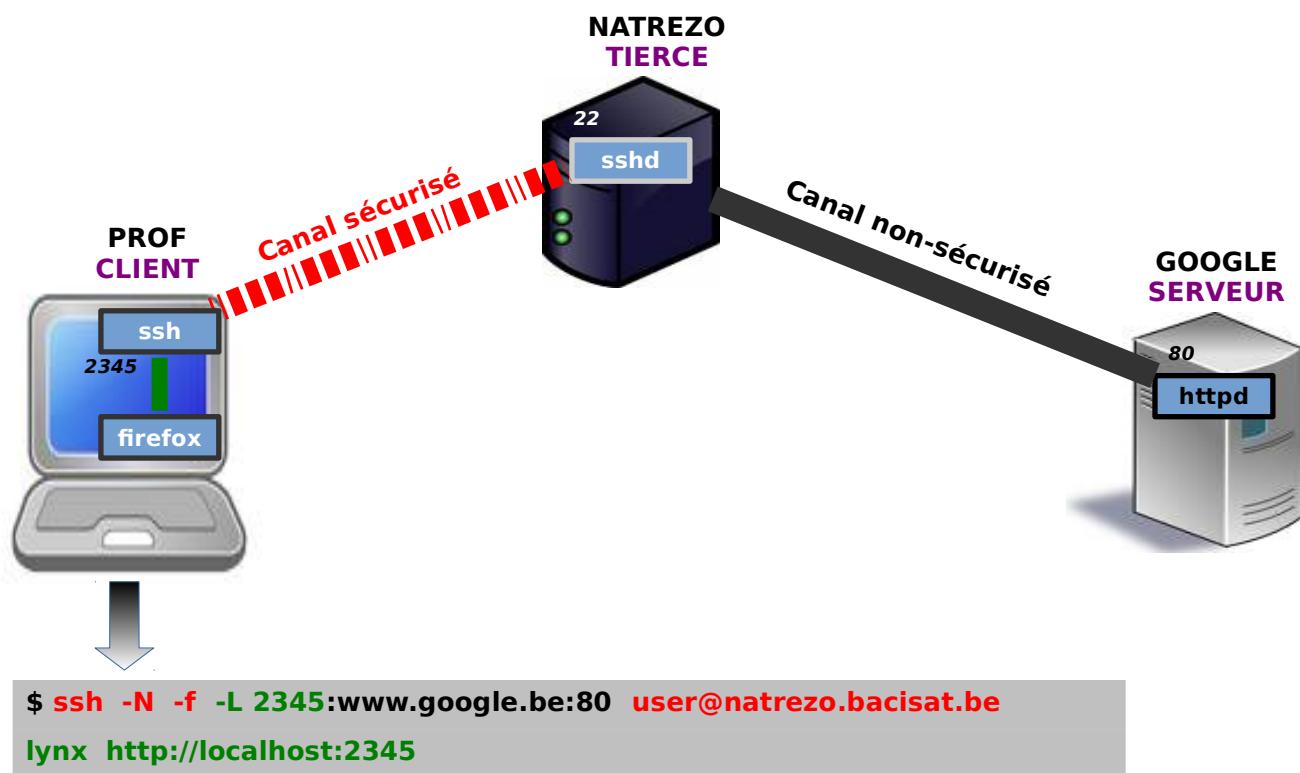
Local - Modèle à 2 machines - Atelier

- Après création du canal:
 - relevez la présence de celui-ci sur les 2 machines.
- relevez les ports ouverts sur les 2 machines. Cela correspond-t-il au schéma ?
- Après ouverture de la communication telnet, relevez les ports ouverts sur les 2 machines. Cela correspond-t-il au schéma ?
- Fermez le canal sécurisé.



Le tunneling

Local - Modèle à 3 machines - Soit 'tunneliser' une conversation http...



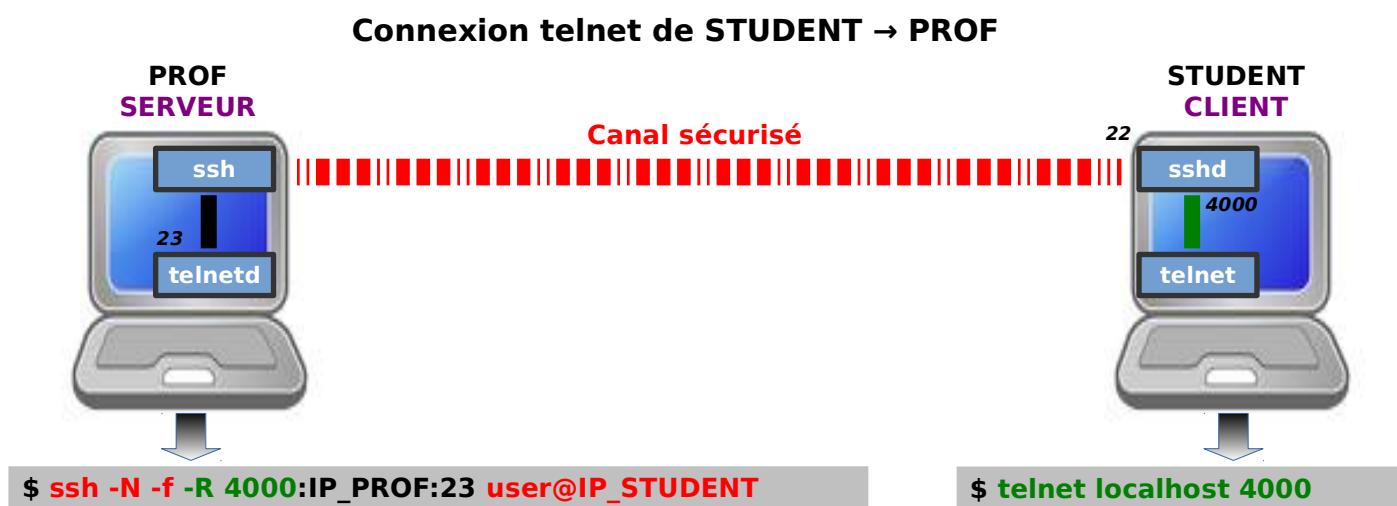
Local - Modèle à 3 machines - Constatations

- Un modèle à 3 machines implique l'utilisation des 2 adresses ip différentes dans la commande.
- Tout fonctionne comme si <http://localhost:2345> pointait directement sur www.google.com:80
- Le client n'a besoin d'aucun daemon serveur sur sa machine mais la création du tunnel et son utilisation est à sa charge. De plus, il a besoin d'un compte sur la machine tierce.
- Ici, le sshd et le service à toucher sont sur des machines différentes.
- La communication est chiffrée seulement entre le client et la machine tierce.



Le tunneling

Remote - Modèle à 2 machines - Soit 'tunneliser' une conversation telnet...



① **Création d'un canal SSH du client SSH vers le serveur SSH ...**

② **Tout ce qui arrivera à ssh par ce canal sera redirigé vers le port 23**

③ **Tout ce qui entrera par le port 4000 sera envoyé dans le canal sécurisé.**



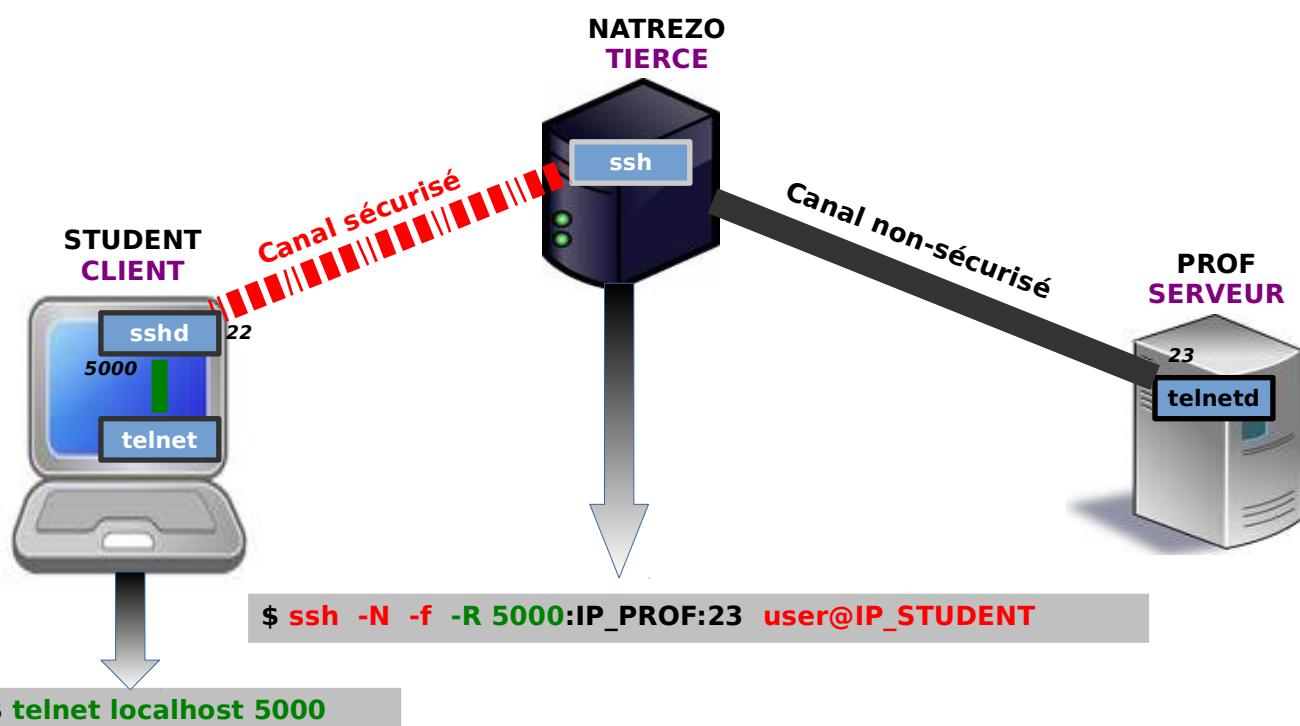
Remote - Modèle à 2 machines - Constatations

- Un modèle à 2 machines implique l'utilisation de 2 adresses ip différentes dans la commande.
- Le canal reste ouvert même après fermeture de la communication telnet.
- Le client a besoin d'un sshd et d'un compte sur sa machine mais la création du tunnel et son utilisation n'est plus à sa charge.
- Le serveur pourrait créer plusieurs canaux vers des clients différents. Ceux-ci n'auraient plus qu'à les utiliser...



Le tunneling

Remote - Modèle à 3 machines - Soit 'tunneliser' une conversation telnet...



Remote - Modèle à 3 machines - Constatations

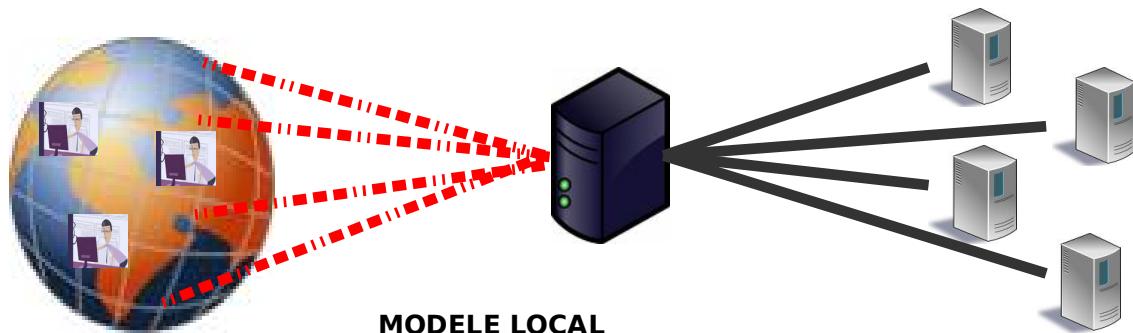
- Un modèle à 3 machines implique l'utilisation des 2 adresses ip différentes dans la commande.
- Tout fonctionne comme si *telnet localhost 5000* pointait directement sur *IP_PROF:23*
- Le client a besoin d'un sshd et d'un compte sur sa machine mais la création du tunnel et son utilisation n'est plus à sa charge. Néanmoins, il n'a pas besoin d'un compte sur la machine tierce.
- Ici, le sshd et le service à toucher sont sur des machines différentes.
- La communication est chiffrée seulement entre le client et la machine tierce.



Le tunneling

Modèle à 3 machines: Remarques

Idéal pour établir, entre plusieurs clients (côté Internet), des communications sécurisées avec plusieurs services d'un Intranet (non chiffré mais considéré comme sûr) via un seul noeud.



Chaque client est à charge de créer son propre tunnel vers la machine tierce sur laquelle il dispose d'un compte Linux.

MODELE REMOTE

La création de tous les tunnels se fait (bien souvent) par l'administrateur de la machine tierce.



- **Autres Remarques**

1° Tunneliser telnet s'avère obsolète car le protocole SSH permet directement d'établir une connexion chiffrée...

2° Pourquoi le n° de port relayé est-il supérieur à 1024 ?

- Pour pouvoir les utiliser par la suite sans être root.
 - Pour que le port sélectionné ne soit pas un port standard déjà en service.
- Conseil: Vérifier quand même si le port relayé est libre.

3° Le client 'Putty' sous Windows permet également de créer des canaux sécurisés ...



Le tunneling

- **Atelier**

Réalisez le tunneling d'une session entre un client et un serveur Ftp.

- 1) Avec le client texte ftp sous Linux.
- 2) Avec le client graphique gftp sous Linux.
- 3) Avec le client graphique FileZilla sous Windows.

Bonus:

- a) Reniflez (*tshark*) votre session sans tunneling et avec tunneling.
- b) Constatez quels sont les ports ouverts sans et avec tunneling .



- **WEBOGRAPHIE**

<http://www.openssh.com>
<http://fr.wikipedia.org/wiki/Ssh>
<http://www.linux-france.org/prj/edu/archinet/systeme/index.html>
https://access.redhat.com/documentation/fr-fr/red_hat_enterprise_linux/7/pdf/system_administrators_guide/Red_Hat_Enterprise_Linux-7-System_Administrators_Guide-fr-FR.pdf

- **BIBLIOGRAPHIE**

SSH, le shell sécurisé - La référence
Par Daniel-J Barrett, Richard-E Silverman
(Edition Oreilly) - *Dispo sur Google books*



Laboratoire de sécurité internet

INTRODUCTION AUX

VPN



Jean-Louis Gouwy



Plan

- Introduction
 - Comment relier 2 sites privés distants ?
 - VPN
- Architectures VPN
 - Architecture physique d'un VPN Client-à-site
 - Architecture physique d'un VPN Site-à-site
 - Le VPN routé
 - Le VPN ponté
- Solutions courantes de mise en œuvre
 - Solutions dédiées
 - Autres solutions
- Le tunnel GRE
 - Généralités
 - Mise en oeuvre
 - L'encapsulation des protocoles
 - Conclusions
 - Exercice 1



- OpenVPN
 - Caractéristiques
 - Avantages et inconvénients
 - Remarques
 - Mise en œuvre
 - . Maquette de travail
 - . Atelier 1 : Configuration à clef partagée
 - . Atelier 2 : Configuration avec authentification par certificat
- Références



Introduction

- **Comment relier 2 sites privés distants ?**

a. Utiliser une ligne louée, proposée par tout bon opérateur de télécoms.

Technologies utilisées: ATM^(*) (*Asynchronous Transfer Mode*), MPLS^(*) (*MultiProtocol Label Switching*) et, plus anciennement, Frame Relay^(*).

Garantie d'un SLA (*Service Level Agreement*) et d'une étanchéité renforcée.

Coût élevé

b. Utiliser l'Internet comme support de connexion.

Technologies utilisées: Les informations seront véhiculées dans un **tunnel**. Différentes technologies (protocoles) permettent de construire des tunnels : PPTP^(*), IPSec^(*), L2TP^(*), GRE, SSL...

. coût réduit
. les FAI peuvent être différents aux extrémités du tunnel.

. l'étanchéité du tunnel n'est pas toujours garantie (ex. GRE)
. la vitesse de transfert des paquets dans le tunnel est limitée au débit de la connexion.



- **VPN**

Les réseaux privés virtuels (*Virtual Private Network, VPN*) sont un moyen de relier deux sites distants à travers un inter-réseau aussi complexe que l'Internet.

L'**intégrité** et la **confidentialité** des données qui transitent entre les 2 sites et l'**authentification** de ceux-ci doit être assurée.

La connexion entre les 2 sites est assurée par un tunnel.

En utilisant un VPN, deux réseaux locaux séparés par un réseau public (Internet) auront l'impression d'être connectés directement.



Introduction

- **VPN**

Nous étudierons 2 solutions de mise en œuvre de tunnels : GRE et OpenVPN

Elles utilisent le principe du tunneling (encapsulation des flux de données circulant entre deux réseaux)

Tunnel GRE : Permet de relier 2 sites mais le transit des données n'est pas sécurisé (pas d'intégrité, ni de confidentialité, ni d'authentification) → le réseau virtuel mis-en-œuvre par ce type de tunnel ne peut pas être qualifié de VPN.

Tunnel OpenVpn : Permet de relier 2 sites avec:

- non sécurisation du transit (comme un tunnel GRE)
- une sécurisation du transit : intégrité, confidentialité, [authentification par certificat].



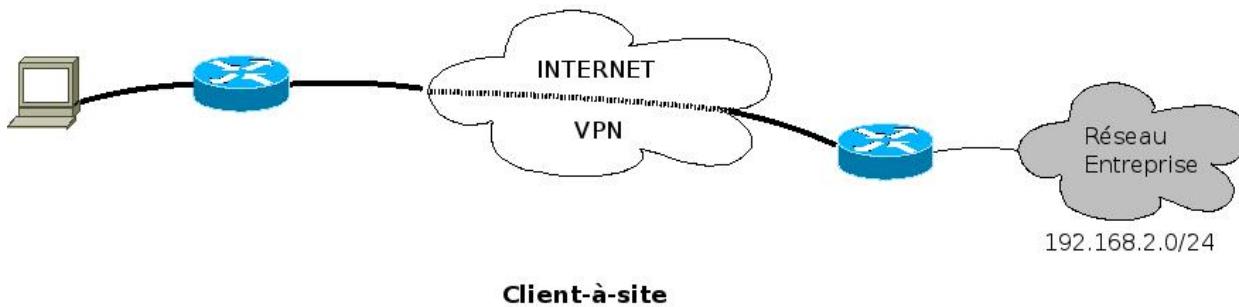
Présentation ludique de ce qu'est un VPN

<https://www.frameip.com/vpn/?video=205#video-205>

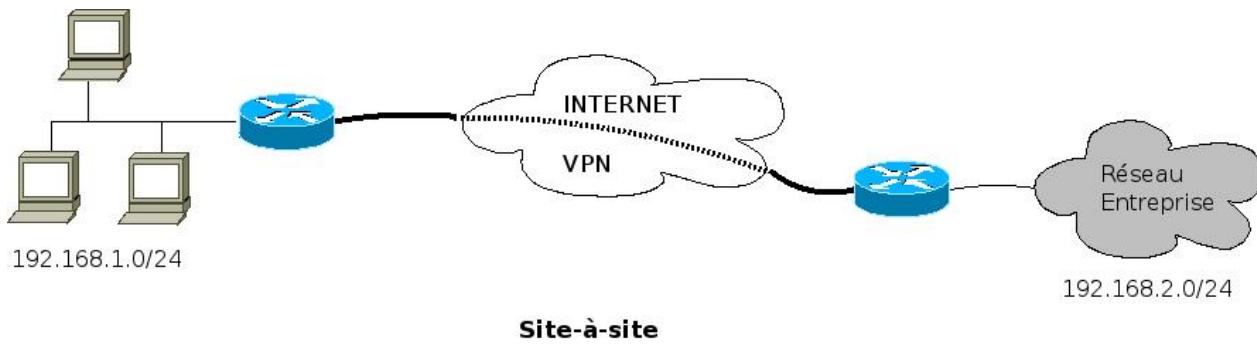


Architectures VPN

- **Architecture physique d'un VPN Client-à-site**



- **Architecture physique d'un VPN Site-à-site**



7

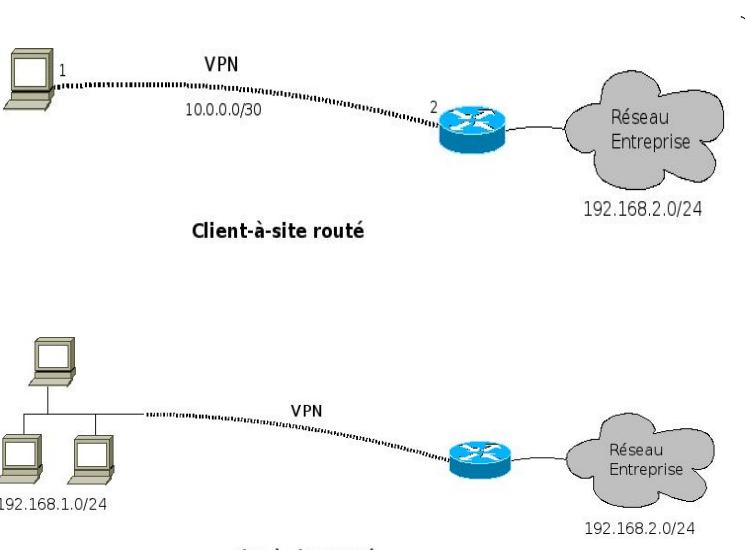


Architectures VPN

- **Le VPN routé**

Les 2 sites auront l'impression d'être connectés entre eux par un seul routeur.

Architectures logiques résultantes :



😊 Solution la plus simple à mettre en œuvre.

😢 Puisque virtuellement un routeur se trouve entre les 2 réseaux, il ne sera possible de faire transiter que des paquets routables (IP) dans le VPN.

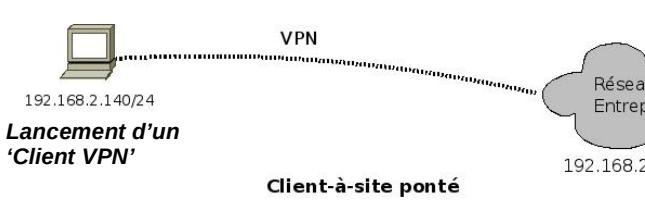
Le broadcast (et multicast) ne peuvent pas transiter d'un réseau à l'autre.



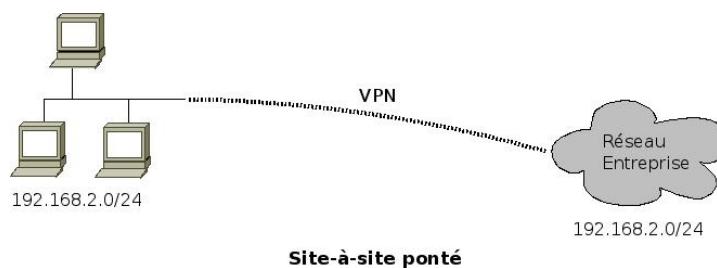
- **Le VPN ponté**

Les 2 sites auront l'impression d'être sur le même réseau (les 2 réseaux auront la même adresse réseau).

Architectures logiques résultantes :



😊 Solution moins évidente à mettre en œuvre.



😢 Solution plus complète car, peuvent transiter :

- tout type de paquets de niveau 3 (IP, IPX...).
- Le broadcast (et le multicast).

VPN (JL Gouwy)

9

Solutions courantes de mise en oeuvre

- **Solutions dédiées**

Utilisation d'équipements orientés VPN et sécurité (pare-feu Checkpoint, Cisco ASA ...)

- **Autres solutions**

- Utilisation du protocole IPsec (solution préconisée par RHEL).
- Utilisation du protocole OpenVPN (solution utilisant la bibliothèque SSL).
- ...

VPN (JL Gouwy)

10

- **Généralités**

- . GRE (Generic Routing Encapsulation)
- . Protocole initialement développé par CISCO multi-plateformes (RFC 2784)



- Possibilités d'ouvrir plusieurs tunnels depuis un hôte donné.
- Permet d'encapsuler n'importe quel protocole de niveau 3 dans IP (souvent de l'IP dans de l'IP).



Ce n'est pas un protocole sécurisé.

- . Pas de chiffrement des données qui passent dans le tunnel (il n'est pas étanche).
- . Pas d'authentification (on n'est pas certain de l'identité du bout du tunnel)

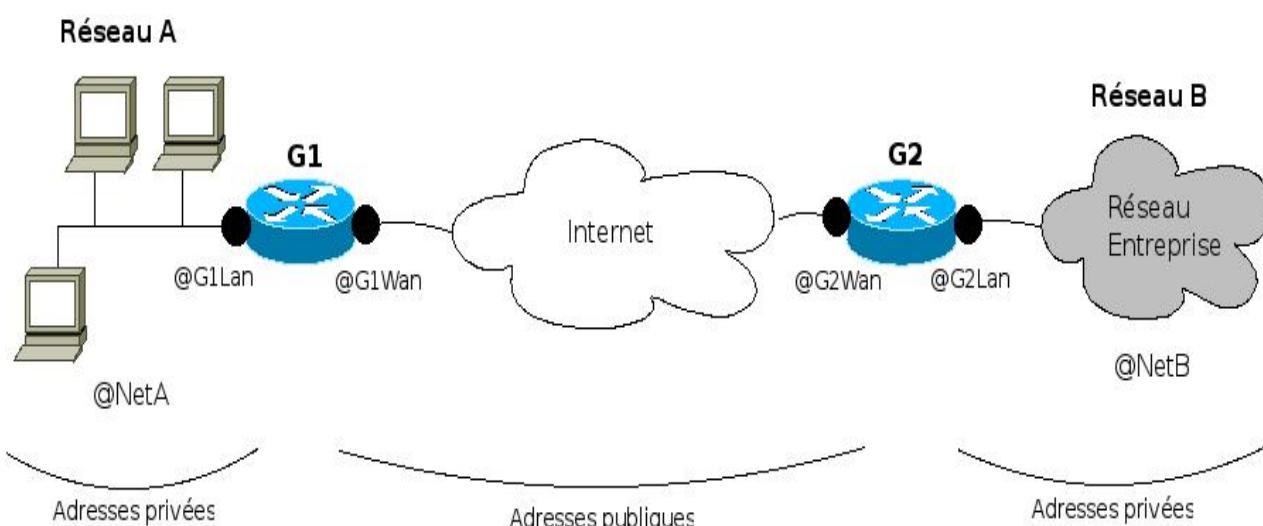
→ le réseau virtuel mis-en-œuvre par ce type de tunnel ne peut pas être qualifié de VPN.



Le tunnel GRE

- **Mise en oeuvre**

- ➊ Soit l'architecture physique suivante



- **Mise en oeuvre**

❷ Création du tunnel

G1# modprobe ip_gre → chargement du module qui permettra la gestion du GRE (encapsulation GRE et IP...)

G1# ip tunnel add tonetb mode gre remote @G2Wan local @G1Wan ttl 255
 → création de l'interface de nom tonetb qui va conduire au réseau B en renseignant les @Ip publiques des extrémités du tunnel.

G1# ip link set tonetb up → montage de cette interface

G1# ip addr add @G1Lan dev tonetb → lui attribuer l'@de la passerelle physique (côté Lan)

G1# ip route add @NetB dev tonetb → baliser la route vers le réseau B

Et configurer l'autre bout du tunnel ...

G2# modprobe ip_gre

G2# ip tunnel add toneta mode gre remote @G1Wan local @G2Wan ttl 255

G2# ip link set toneta up

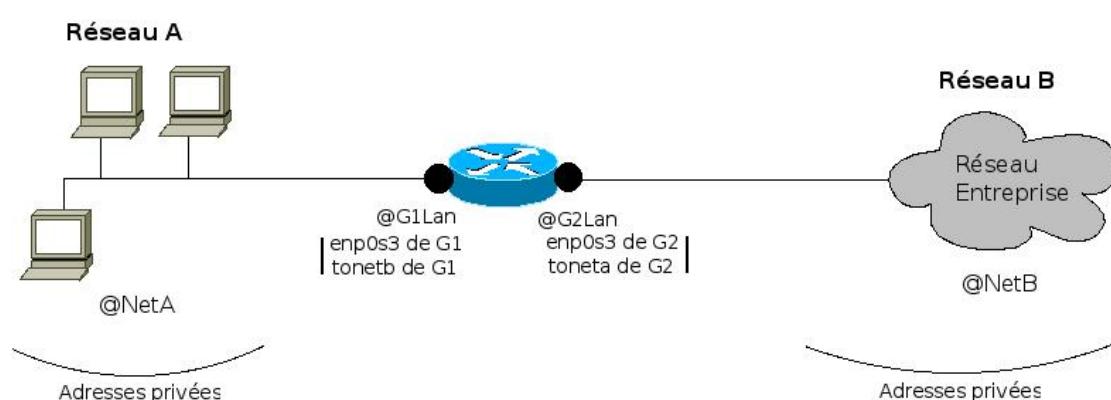
G2# ip addr add @G2Lan dev toneta

G2# ip route add @NetA dev toneta



- **Mise en oeuvre**

❸ Architecture logique générée



En utilisant leur @Ip privées, n'importe quelle machine du ‘Réseau A’ peut communiquer tout autre machine du réseau B (et inversement).

Tout cela en passant par l’Internet !!!

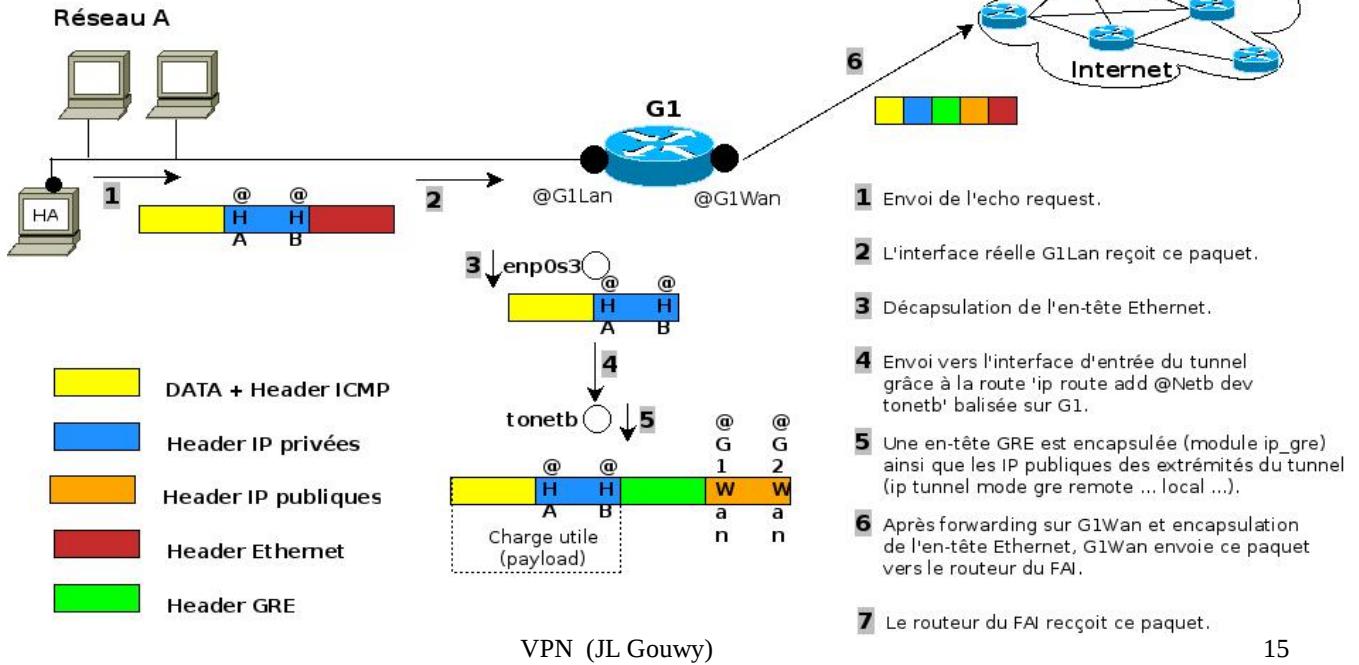


Le tunnel GRE

- L'encapsulation des protocoles

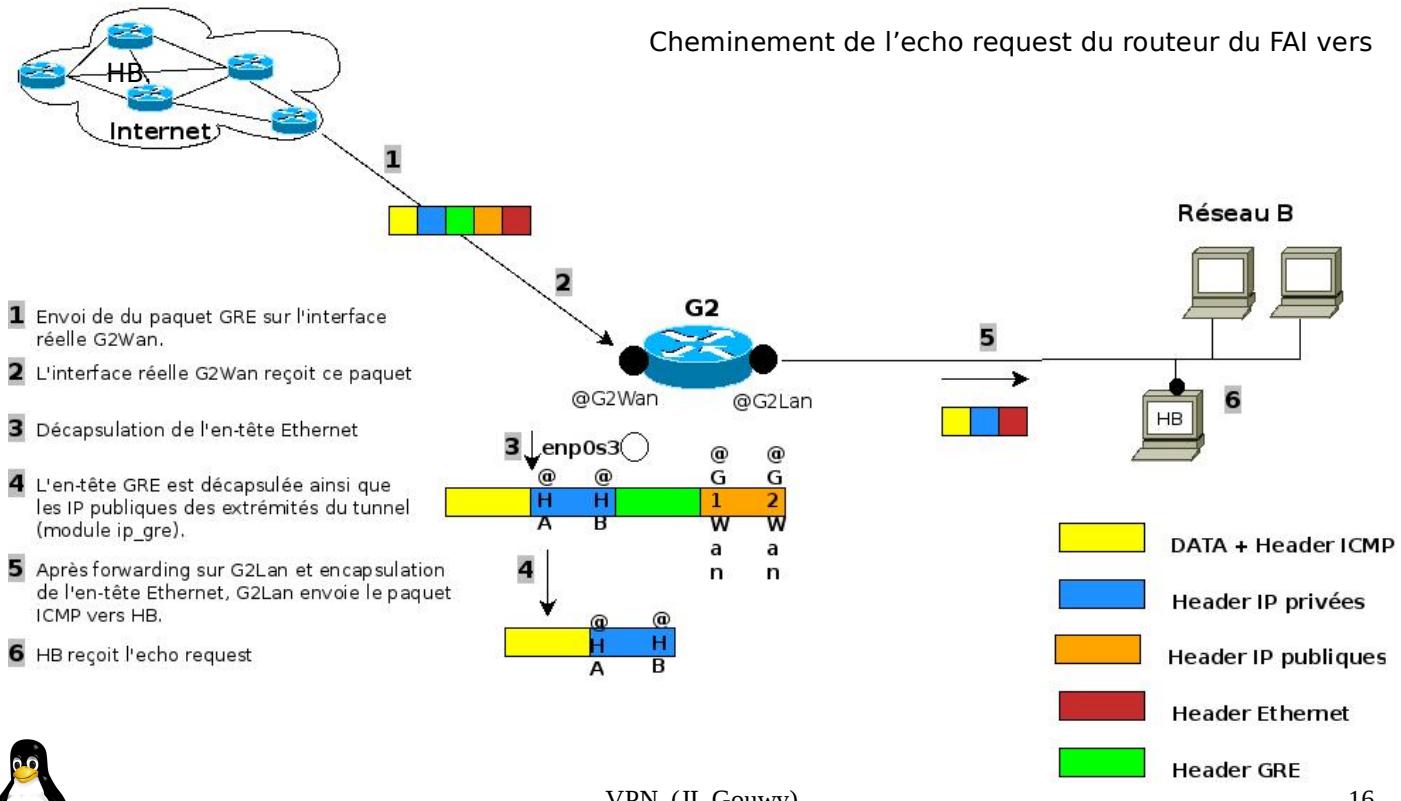
Soit un host du 'Réseau A' (HA) envoyant un ping vers un host du 'Réseau B' (HB).

Cheminement de l'echo request de HA vers le routeur du FAI



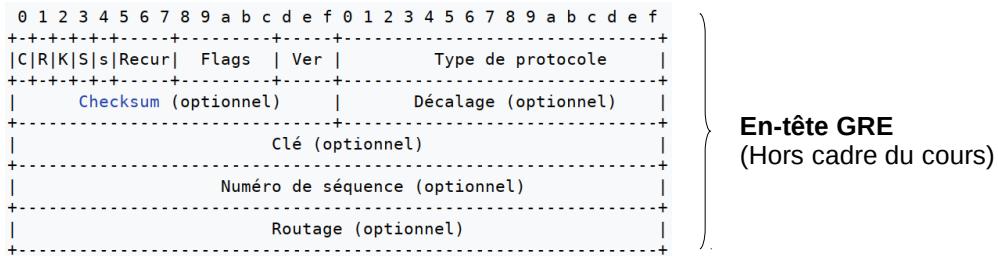
Le tunnel GRE

- L'encapsulation des protocoles



- **Conclusions**

- . GRE est un protocole de niveau supérieur qui sera routé par de l'IP. Il doit donc transporter des données (ici le payload = icmp + ip privées) au même titre que TCP, UDP, ICMP...



- . GRE va véhiculer des paquets de niveau 3. Ici, il s'agit de paquets IP. Mais tout autre type de paquets (IPX,...) serait également pris en charge.
- . GRE permet d'ouvrir depuis un hôte donné autant de tunnels que l'on désire vers différents réseaux distants.
- . GRE : Solution peu sécurisée. Il faudrait au moins chiffrer les données avant de les envoyer dans le tunnel...

→ Remède:

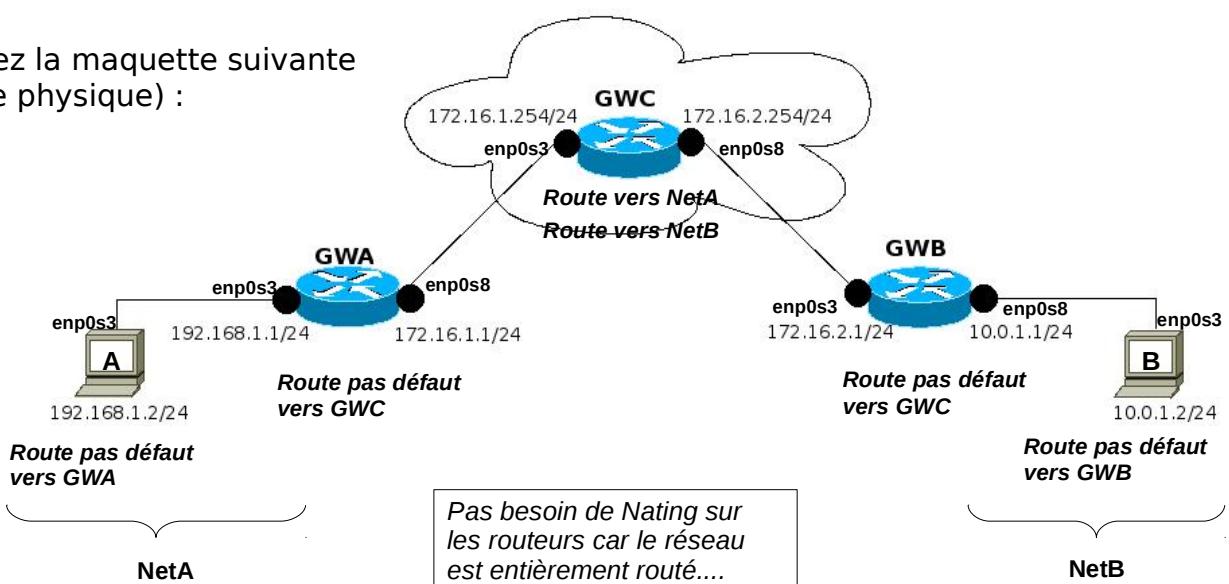
IPSec: Solution plus sécurisée mais plus délicate à mettre en œuvre.



Le tunnel GRE

- **Exercice 1**

- a. Créez la maquette suivante (vue physique) :

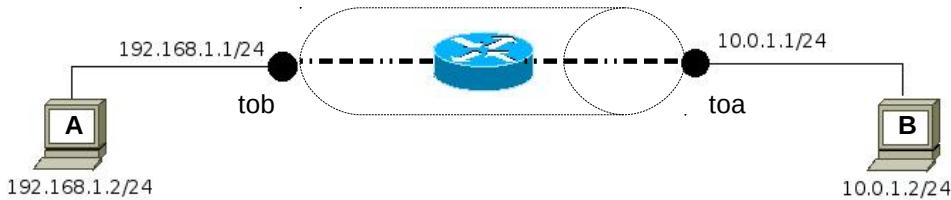


L'exercice est réalisé dans un laboratoire de machines virtuelles. Ainsi, les adresses des passerelles VPN sont des adresses privées. Cependant, en réalité, elles seront publiques.



- **Exercice 1**

- b. Mettez en œuvre le tunnel GRE suivant (vue logique) pour que les machines A et B puissent se toucher à travers le tunnel:



Le tunnel GRE

- **Exercice 1**

- c. Vérifiez la configuration des interfaces et des tables de routage sur GWA et GWB.
d. Vérifiez par un ‘traceroute’ que les 2 hôtes se joignent par le tunnel.
e. Capturez, sur l’Internet, une trame tunnelisée et émise par un ping de A vers B.
Vérifiez qu’elle est bien composée de 5 couches (voir la théorie sur l’encapsulation).
Montrez que les data générées par l’echo request transitent en clair.
f. Déconfigurez votre tunnel.
Un ping de A vers B fonctionne-t-il toujours ? Pourquoi ?
Que se passerait-il en réalité ?
g. Configurez votre maquette pour que le tunnel soit permanent.



- **Caractéristiques**

- . C'est une application client/serveur capable de créer des tunnels (routés ou pontés) le plus souvent entre 2 réseaux distants.
- . C'est aussi le nom du protocole (basé sur SSL/TLS) qui en découle.
- . Les tunnels peuvent être:
 - non sécurisés (au même titre que les tunnels GRE)
 - sécurisés par:
 - clé secrète partagée ou
 - certificat électronique ou
 - login/password
- . Port écoute par défaut: UDP 1194

→ si la mise-en-œuvre du tunnel est sécurisée, le réseau virtuel sous-jacent peut être qualifié de VPN.



OpenVPN

- **Avantages et inconvénients**



- . Opensource et fiable.
- . Gratuit au niveau logiciel.
- . Multi-plates-formes.
- . Très répandu et facile à mettre en place.
- . Complètement transparent pour l'OS.
- . Capable de tourner sur n'importe quel port en écoute côté serveur.
- . Un port unique peut être utilisé pour plusieurs tunnels sur le serveur VPN.



- . Le protocole OpenVPN n'est pas pris en charge nativement par les OS
 - nécessité d'installer un logiciel OpenVPN.
- . Moins efficace qu'IPSec



- **Remarques**

Beaucoup d'autres protocoles permettant de créer des tunnels VPN existent :

PPTP: En natif sur la plupart des plateformes.
Pas besoin d'installer un autre logiciel.
Il n'est pas réputé fiable.

L2TP/Ipsec: En natif sur la plupart des plateformes.
Plus lent que OpenVPN.
Utilisation de plusieurs ports fixes
→ peut se retrouver bloqué sur un réseau où certains ports ne sont pas autorisés.

SSTP: Chiffrements forts et évite la plupart des pare-feu.
Intégré à Windows seulement (pas open source).

IKEv2: Rapide, supporte des chiffrements forts et fournit une connexion très stable.
Il n'est pas supporté par toutes les plateformes.
Comme L2TP/Ipsec, il utilise plusieurs ports fixes.

...



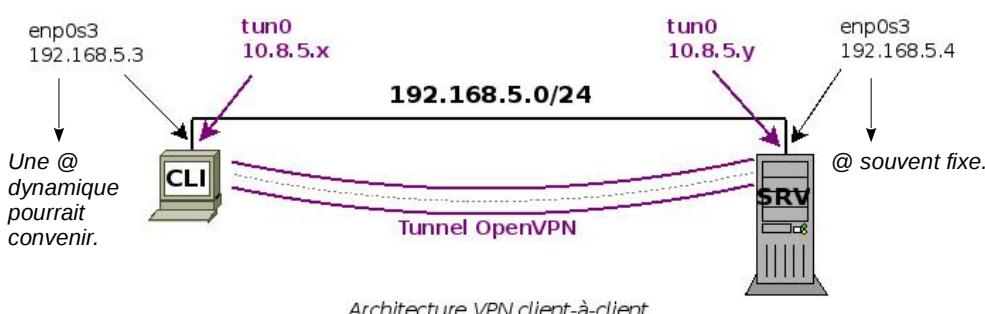
OpenVPN

- **Mise en oeuvre**

Maquette de travail

Une machine héberge le serveur VPN, un autre le client VPN.

Dans cette maquette, que la machine soit connectée via internet ou non change rien.
Nous allons configurer le client et le serveur sur le même réseau, et établir un tunnel OpenVPN entre eux.



tun pour ip (tap pour ethernet) sont des pilotes permettant de créer les cartes virtuelles tun0, ... devant se trouver à chaque extrémité du tunnel.

Nous constaterons que les données transitant par le tunnel seront sécurisées; ce qui ne sera pas le cas pour celles passant par le réseau local.



Atelier 1 : Configuration à clef partagée

Etapes à suivre:

1. Installation du package openvpn sur le serveur et le client.
2. Génération de la clef partagée sur le serveur et rapatriement de celle-ci sur le client.
3. Mise en œuvre manuelle du tunnel à l'aide de cette clef.
4. Contrôles.
5. Mise en œuvre du tunnel en mode service.
6. Autres options possibles.

Conclusion



OpenVPN

Atelier 2 : Configuration avec authentification par certificat

Etapes à suivre:

1. Installation d'une autorité de certification sur le serveur (PKI). Il y aura donc création d'un certificat racine CA (autorité de certification) qui servira à signer d'autres certificats.
2. Création d'un certificat et de la clé privée du serveur VPN et configuration du serveur VPN. Pour des raisons pratiques, le serveur PKI et le serveur VPN seront sur la même machine.
3. Création d'un certificat et la clé privée du client VPN et configuration du client VPN. Le certificat client devra être transféré du serveur PKI vers la machine abritant le client VPN.
4. Implémentation sur le client.
5. Contrôles.
6. Etablissement du tunnel openvpn
7. Conclusion



- **WEBOGRAPHIE**

<https://www.frameip.com/vpn/>

<http://irp.nain-t.net/doku.php/280vpn:start>

<https://openclassrooms.com/fr/courses/2939006-protegez-l-ensemble-de-vos-communications-sur-internet/2940511-quest-ce-quun-vpn-virtual-private-network-et-a-quoi-sert-il>

- **BIBLIOGRAPHIE**

Red Hat Enterprise Linux/CentOs
Mise en production et administration de serveurs (2ième édition)
Thibault Bartolone - ENI Edition

Les VPN
Fonctionnement, mise en oeuvre et maintenance (2ième édition)
Jean-Paul ARCHIER - ENI Edition

