R4.01 – Infrastructures de sécurité

Antoine Pernot – antoine.pernot@iut-dijon.u-bourgogne.fr

27 – 29 avril 2026

Relecteurs

Arnaud Bitterlin
Ingénieur Réseaux et Télécommunications
IUT R&T Colmar
Université de Technologie de Troyes

Loïc Defief
Ingénieur Réseaux et Télécommunications
IUT R&T Dijon-Auxerre
Université de Technologie de Troyes

Introduction et objectifs

Les objectifs de ce cours sont d'identifier les menaces de sécurité d'un réseau, de mettre en place et de configurer une architecture réseau sécurisée en faisant des choix d'outils adaptés aux besoins.

Les éléments suivants seront abordés :

— Les différentes types d'attaques
— Les bonnes pratiques utilisateur et pour les serveurs
— Les certificats et les infrastructures à clefs publiques
— Les serveurs proxy
— Le pare-feu et les ACL
— Le VPN
— La mise en place d'architectures réseau sécurisées

Les supports de cours, TD et TP, ainsi que des ressources complémentaires sont disponibles sur https ://rtaux.antoinepernot.fr

L'évaluation des travaux pratiques sera effectuée par des tests automatiques qui seront exécutés le 3 mai à 23 heures. Toute opération effectuée passée ce délai ne sera pas prise en compte.

L'évaluation de ce module est répartie comme suit :

— Travaux pratiques : 30%
— Examen final : 70%

Une question, remarque ou suggestion ? Vous pouvez me contacter par courriel à l'adresse : antoine.pernot@iut-dijon.u-bourgogne.fr

Table des matières

Cours

1 Généralités

Les systèmes d'information (SI) sont de plus en plus incontournables dans nos sociétés. Il est indispensable d'en garantir le bon fonctionnement au regard de leur criticité dans notre vie quotidienne. Les interconnexions entre les différents systèmes permises par le développement d'Internet les rendent interdépendant les uns des autres et vulnérables aux attaques.

Pour parer à cela, il est nécessaire de déployer une politique de **Sécurité des Systèmes d'Information (PSSI)**. Cela se traduit par **la mise en place de moyens humains, techniques, organisationnels et juridiques pour garantir, conserver et rétablir la sécurité d'un système, notamment en termes de disponibilité, d'intégrité et de confidentialité**.

Il existe plusieurs types de vulnérabilités au sein d'un SI :

Les vulnérabilités humaines C'est le maillon le plus faible de la sécurité du SI. L'être humain est sujet à des erreurs, des négligences, des incompétences qui peuvent mener à des failles de sécurité exploitables.

Les vulnérabilités technologiques Il s'agit dans les SI d'erreurs de conception, de développement, de configuration, de maintenance, etc.

Les vulnérabilités organisationnelles Ce sont essentiellement des manques de procédures, de documentations ou de circuits de validation pour faire face aux incidents de sécurité (sauvegarde, escalade de problèmes, etc.)

Un SI est soumis à plusieurs types de menaces :

Les menaces naturelles Incendies, inondations, séismes, épidémies, météo, etc.

Les menaces humaines Vol d'informations, sabotage, chantage, usurpation d'identité, espionnage, grèves, etc.

Les menaces juridiques Les réglementations pouvant régir votre SI (notamment dans le cas d'un SI sur plusieurs pays)

Ces menaces peuvent porter sur un ou plusieurs indicateurs du SI (**indicateurs DICP**) :

Disponibilité S'assurer que le service soit fonctionnel lorsque les utilisateurs en ont besoin. La disponibilité se mesure en divisant la durée de fonctionnement effectif par la durée de fonctionnement prévu. Par exemple, une disponibilité minimale de 99,999 % ne permet que 5 minutes et 15 secondes d'indisponibilité.

Intégrité Les données traitées doivent être celles attendues, sans altération ou destruction volontaire ou accidentelle.

Confidentialité Les données doivent être accessibles uniquement aux personnes dont l'accès est autorisé.

Preuve ou Traçabilité Il doit être possible de remonter de manière sûre et fiable les origines d'un évènement. Cela inclut également la non-répudiation : un utilisateur ne peut pas nier avoir effectué une opération ou reçu une information.

2 Les attaques

On peut distinguer trois cibles d'attaques visant un SI :

— Attaques sur les clients (ordinateur, smartphone, tablette, objets connectés, etc.)

— Attaques sur les serveurs

— Attaques sur le réseau (équipements réseaux ou sur les paquets en eux-mêmes)

On désigne par **exploit** l'ensemble des techniques permettant d'exploiter une faille dans un SI. On retrouve ici les injections de code, les dépassements de pile mémoire, les injections SQL, les XSS (cross-site scripting) etc.

2.1 Les logiciels malveillants

Le terme **malware** en anglais désigne les logiciels malveillants dans le but de nuire à un SI. On peut distinguer différents types de malwares :

Virus Le virus s'insère dans un logiciel légitime (nommé hôte) et se propage vers d'autres ordinateurs (*via* le réseau ou les périphériques de stockage externes).

Ver Le ver (**worm** en anglais) est similaire au virus à la différence qu'il n'a pas besoin d'un programme hôte pour se propager.

Cheval de Troie À l'instar de cet épisode de la mythologie grecque, le cheval de Troie (**Trojan horse** en anglais) informatique dissimule ses mauvaises intentions sous une apparente légitimité. L'utilisateur dupé l'installe sur son ordinateur et permet au code malveillant de s'exécuter. Ils peuvent prendre la forme d'un document, d'un logiciel légitime, etc.

Rootkit Ce terme désigne les techniques mises en œuvre par un pirate pour maintenir un accès frauduleux sur une machine.

Les impacts de ces malwares peuvent être très variés :

— Vol d'informations

— Saturation des ressources des ordinateurs ou du réseau

— Utilisation frauduleuse de ressources (par exemple minage de cryptomonnaies)

— Enrôlement dans un botnet : un réseau d'ordinateurs infectés dans le but d'effectuer une attaque sur une autre cible (en général de l'envoi de spam ou une attaque par déni de service distribué).

— Altération ou destruction de données

— Destruction du matériel

— Chantage (**rançongiciel** ou **ransomware** en anglais)

— etc.

Certains malwares sont conçus pour une cible très spécifique dans le cas d'attaques organisées (le ver Stuxnet, développé par la NSA et découvert en 2010 visait les automates industriels Siemens utilisés dans les centrifugeuses d'enrichissement d'uranium en Iran).

2.2 L'ingénierie sociale

Il s'agit de **techniques de manipulation psychologique** visant à exploiter les faiblesses de l'utilisateur d'un SI. Elles exploitent les faiblesses psychologiques, sociales et organisationnelles des personnes ou des organisations dans le but d'obtenir frauduleusement quelque chose.

Les objectifs de ce type d'attaques sont très variés :

— Escroquerie

— Vol d'informations

— Propagation de malwares

— Accès au SI

— etc.

Les méthodes d'attaques sont très variées : on connaît surtout les courriels ou sites frauduleux (**hameçonnage** ou **phishing** en anglais), mais cela peut passer également par des appels téléphoniques, des SMS, des courriers, des affiches, etc.

Une **bonne sensibilisation et formation** des utilisateurs du SI permettent de limiter les risques de ce type d'attaques. Cependant, les attaques d'ingénierie sociales peuvent être spécialement conçues pour viser une organisation, ce qui les rend plus difficilement identifiables par les utilisateurs.

2.3 L'exploitation de failles

On retrouve ici différentes techniques visant à utiliser les failles d'un système (matériel, système d'exploitation, site Web, services, etc.) à des fins malveillantes. Voici quelques exemples des techniques employables :

Injections SQL Insérer dans une requête SQL légitime du code malveillant.

Cross-site scripting Abrégé XSS pour *Cross-site scripting* en anglais. Injecte du contenu dans une page Web en vue d'être exécuté sur le navigateur de la ou des victimes.

Débordement de tampon Un bug écrit dans une zone mémoire à l'extérieur de la zone prévue et permet à un attaquant d'exécuter du code malveillant.

2.4 Les attaques du réseau

Sont regroupées sous cette appellation l'ensemble des techniques visant à attaquer un SI par son réseau. On retrouve les attaques suivantes :

Sniffing Cette technique consiste à écouter le trafic réseau en vue de collecter des informations (mots de passe, données, etc.)

Usurpation d'adresse IP ou MAC Utilisée par un attaquant visant à se faire passer pour une autre machine en vue de collecter des informations ou de fournir de fausses informations à un client (empoisonnement DNS, etc.)

Déni de service Dans une attaque par déni de service (**denial of service (DoS)** en anglais), l'attaquant cherche à saturer les réseaux ou les serveurs cibles afin de rendre le service inopérant. Si on utilise plusieurs sources pour effectuer l'attaque, comme dans le cas de l'utilisation d'un botnet, on parle de **déni de service distribué** (**distributed denial of service (DDoS)** en anglais).

3 Bonnes pratiques

À tous les niveaux d'un SI, tant dans le milieu professionnel que personnel, il est important d'adopter des bonnes pratiques afin de garantir un bon niveau de sécurité et de se prémunir des attaques. En voici quelques unes qu'il est bon d'adopter et de partager à l'ensemble des collaborateurs de votre organisation et à votre entourage.

- Former et sensibiliser les parties prenantes du réseau. Les équipes opérationnelles ayant un accès privilégié au réseau doivent suivre des formations particulières liées à leurs droits forts, ainsi qu'à leurs responsabilités. Chaque utilisateur est un maillon essentiel de la sécurité du SI.
- Effectuer un inventaire précis de votre SI : les serveurs, le schéma du réseau, les comptes utilisateurs, etc.
- Prévoir des procédures d'arrivée, de départ et de mutation claires pour fournir les bonnes permissions aux bonnes personnes au bon moment.
- N'autoriser sur le réseau uniquement les équipements maîtrisés.
- Verrouiller et attacher le poste lorsque vous n'êtes pas présent.
- Ne pas connecter de périphérique d'origine inconnue (clef USB, disque dur, etc.).
- Identifier nommément chaque personne accédant au SI avec des droits restreints uniquement aux besoins des utilisateurs. Ne fournissez pas de compte générique et retirez les droits administrateurs au maximum d'utilisateurs.
- Choisir des mots de passe forts, uniques pour chaque service et individuels. L'utilisation d'un coffre-fort de mots de passe (exemple : KeePass) permet de stocker les mots de passe de manière sécurisée. L'utilisateur n'a plus qu'à retenir le mot de passe du coffre. Ne stocker ni ses mots de passe en clair, ni sur un document papier accessible.

Un mot de passe est comme un slip : c'est personnel, ça se change souvent et on ne le laisse pas traîner sur son bureau !

— Changer les éléments d'authentification par défaut des équipements et des services.

— Activer dès que cela est possible l'authentification forte.

— Sécuriser les équipements du réseau (pare-feu, anti-virus, limiter les applications et extensions installées à l'essentiel, chiffrer les données sensibles, etc.)

— Sauvegarder régulièrement sur un système hors ligne les données importantes.

— Limiter l'utilisation des WiFi publics et si cela est nécessaire, chiffrer les communications avec un VPN.

— Chiffrer les communications (notamment sur Internet) et choisir les protocoles sécurisés quand cela est possible.

— Cloisonner le réseau en zones distinctes en fonction des usages (bureautique, DMZ, etc.). N'exposer que les services nécessaires sur Internet.

— Mettre en place une passerelle d'accès sécurisé à Internet (pare-feu, proxy).

— Choisir un système de sécurité du WiFi robuste et segmenter les différents réseaux (visiteurs, bureautique, etc.)

— Sécuriser l'accès physique aux salles serveurs et locaux techniques.

— Utiliser un réseau dédié et couper l'accès à Internet aux postes et serveurs utilisés pour l'administration du SI.

— Sécuriser les connexion réseau pour les équipements nomades (VPN).

— Appliquer les mises à jour des logiciels sur l'ensemble du SI.

— Journaliser les événements sur les serveurs et équipements réseau du SI.

— Contrôler régulièrement l'application des règles de sécurité.

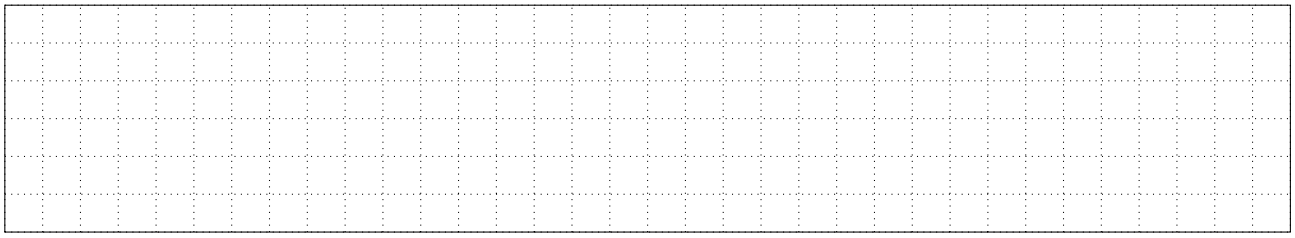
— Définir une procédure en cas d'incident de sécurité.

Ces quelques préconisations sont issues du guide d'hygiène informatique fourni par l'ANSSI. Le document complet est disponible sur :
https://www.ssi.gouv.fr/guide/guide-dhygiene-informatique/

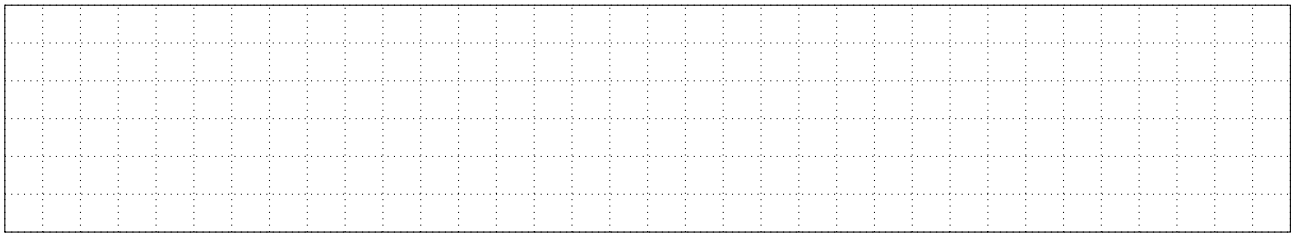
Exercice 1

Pour les situations suivantes, décrivez la posture à adopter afin de garantir la sécurité du SI. Justifiez.

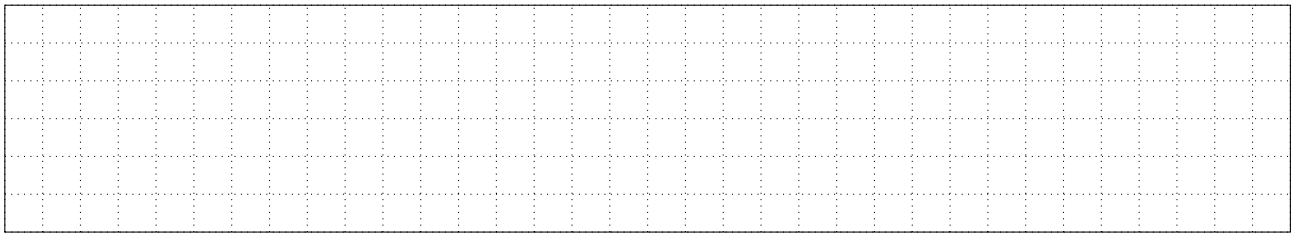
1. Un utilisateur souhaite copier un dossier confidentiel sur une clef USB personnelle pour le terminer chez lui.



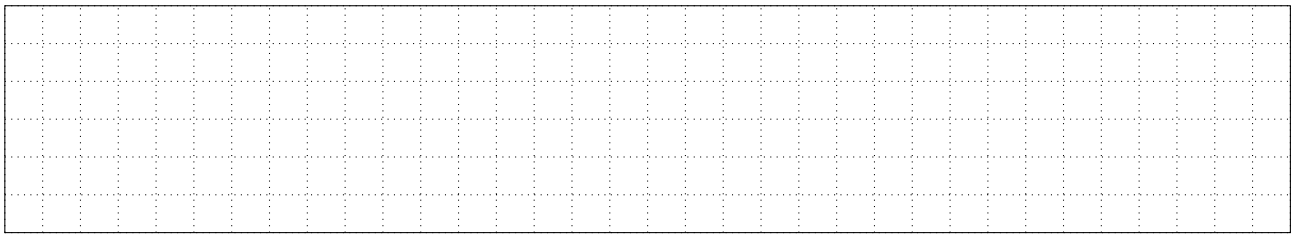
2. Un utilisateur veut transférer des données sensibles sur un service de partage de fichiers public.



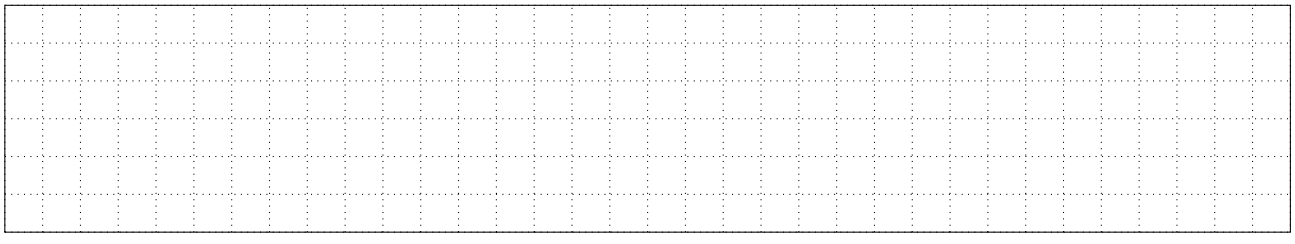
3. Une utilisatrice souhaite enregistrer ses mots de passe dans un coffre à mots de passe Keepass.



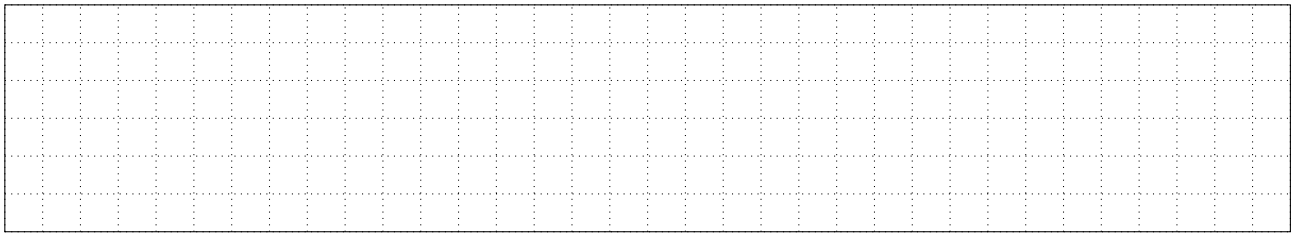
4. Le compte d'une directrice ayant démissionné depuis trois mois est encore actif et utilisable à distance.



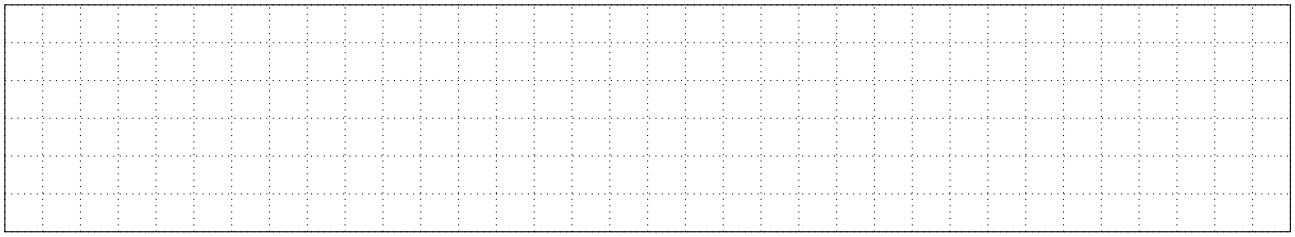
5. Une équipe de développement souhaite connecter directement leur serveur de tests sur le cloud avec l'annuaire interne.



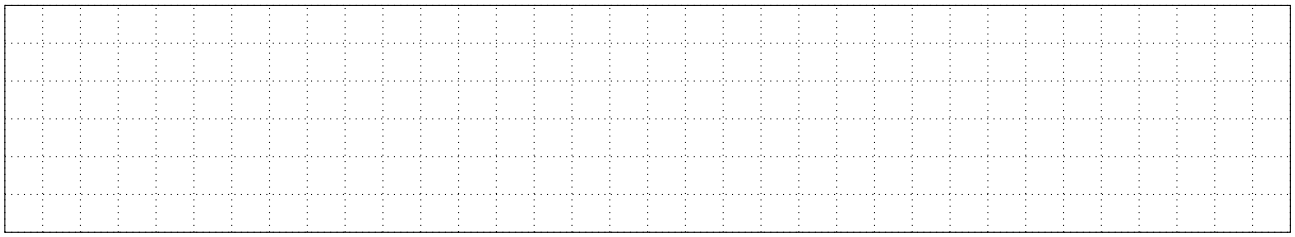
6. Un commercial s'est fait dérober son ordinateur portable lors d'un déplacement.



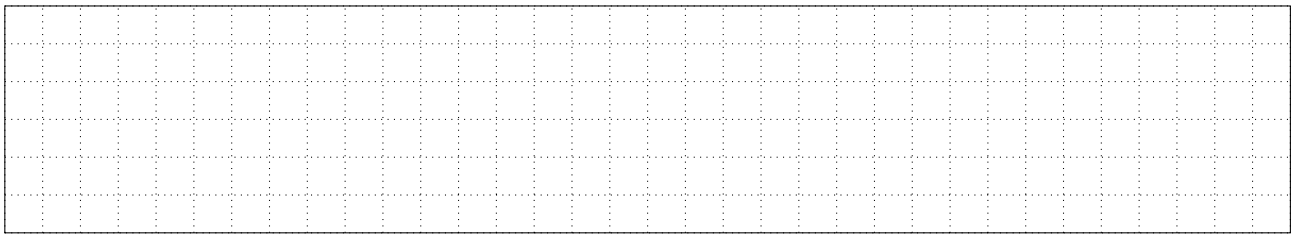
7. Votre directrice générale souhaite offrir l'accès à Internet aux visiteurs de votre société.



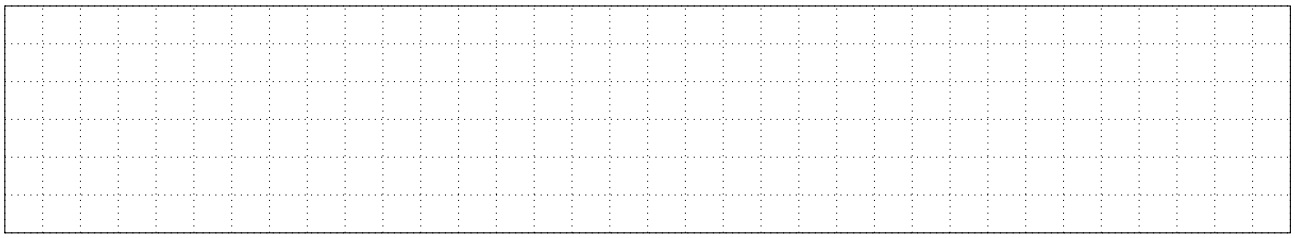
8. Une commerciale souhaite avoir les accès administrateur sur son poste pour pouvoir résoudre ses incidents bureautiques elle-même.



9. Le service marketing souhaite désactiver le proxy car ce dernier les bloque régulièrement dans leurs travaux.



10. Un commercial souhaite pouvoir se connecter depuis le Wifi des hôtels.



4 Le chiffrement et la signature cryptographique

Les procédés de signature et de chiffrement cryptographiques permettent d'apporter respectivement **l'intégrité d'un message** et **la confidentialité d'un message**.

Les standards de chiffrement et de signature les plus courants se basent sur la **cryptographie asymétrique**.

Le fonctionnement, décrit ici pour des messages, fonctionne de la même manière pour la communication entre un client et un serveur d'un service chiffré (HTTPS, SMTPS, etc.).

Contrairement au chiffrement symétrique qui utilise la même clef pour chiffrer et déchiffrer, le chiffrement asymétrique utilise deux clefs :

— **La clef publique** pouvant être diffusée auprès des tiers avec lesquels on communique.

— **La clef privée** devant être gardée secrète.

Ainsi, il est nécessaire à ce que l'expéditeur possède **la clef publique** du destinataire afin de chiffrer ou signer un message lui étant destiné *(fig. 1)*.

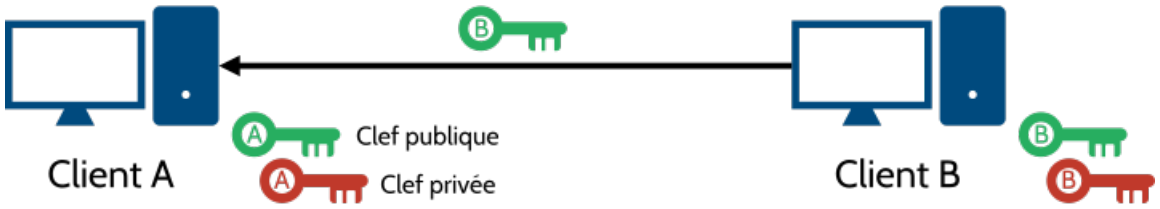


FIGURE 1 – Échange de clef publique de B

Voici la terminologie à employer :

chiffrer : **Transformer un message en clair en un message inintelligible pour assurer le secret de sa transmission.**

déchiffrer : **Traduire en clair un message chiffré à l'aide de la clef de déchiffrement.**

décrypter : **Traduire en clair un message chiffré en ignorant la clef de déchiffrement.**

ATTENTION : Le terme "crypter" reviendrait à chiffrer sans connaître la clef de chiffrement, ce qui n'a aucun sens. L'usage de ce terme est à bannir.

4.1 Le chiffrement

Le chiffrement permet de rendre illisible le contenu du message à toute personne ne possédant pas la clef de déchiffrement (**clef privée** dans le cas d'un chiffrement asymétrique).

L'envoi d'un message chiffré de A vers B se fait comme suit *(fig. 2)* :

1. Le message en clair est **chiffré** à l'aide de **la clef publique de B**.
2. Le message chiffré est envoyé à B.
3. Le message est **déchiffré** par la **clef privée** de B.

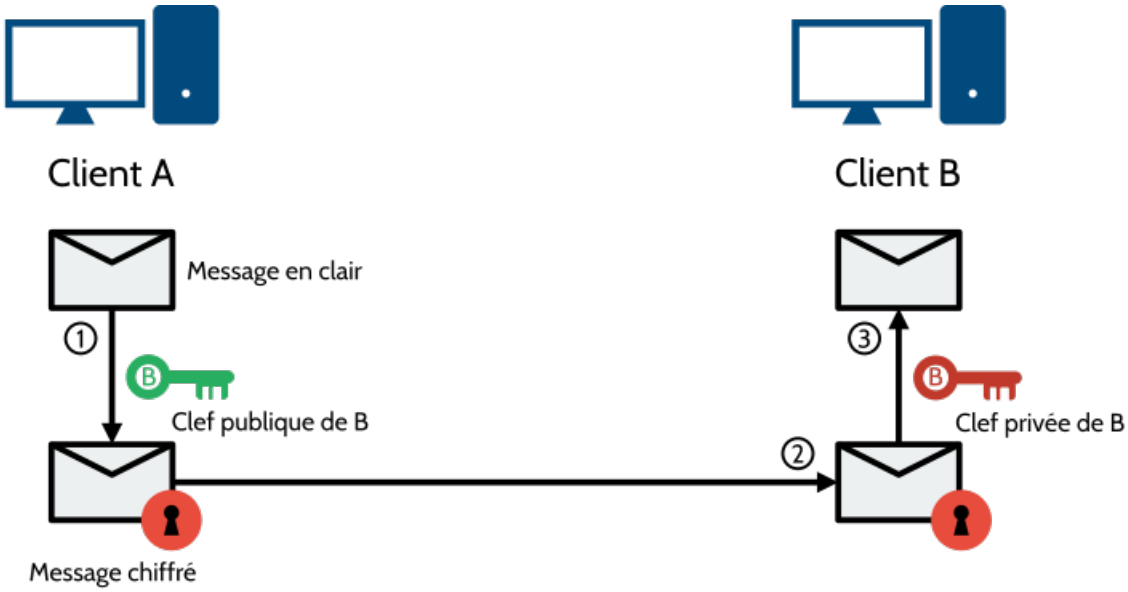


FIGURE 2 – Envoi d'un message chiffré

4.2 La signature numérique

La signature numérique permet de garantir l'origine du message et son intégrité. Elle possède les avantages suivants :

— Elle est **authentique et infalsifiable** : le signataire est identifié et son identité ne peut être usurpée.

— La signature est **à usage unique** : on ne peut pas réutiliser la signature d'un message pour en signer un second.

— Le document est **inaltérable** : s'il est modifié, la signature devient invalide.

— La signature est **irrévocable** : le signataire ne peut pas nier la signature vu qu'il est seul à posséder la clef privée.

Ainsi, la signature numérique propose de nombreux avantages par rapport à la signature manuscrite.

L'envoi d'un courriel signé entre A et B s'effectue comme suit *(fig. 3)* :

1. Une **empreinte** du message est calculée à l'aide d'une **fonction de hachage**. Une fonction de hachage produit une **empreinte unique** pour un contenu. Il s'agit d'une **fonction à sens unique** qui ne permet pas à partir de l'empreinte de retrouver le contenu originel. Quelques fonctions de hachages courantes : MD5, SHA1, SHA256, SHA512. Certaines sont à éviter

telles que le MD5 et SHA1 car il y a risque de collision (le fait que deux contenus différents aient la même empreinte).

2. L'empreinte est **chiffrée** avec la **clef privée de l'émetteur**. On obtient la **signature** du message.
3. La signature est ajoutée au message en clair.
4. Le message signé est envoyé.
5. La signature est séparée du message.
6. La signature est **déchiffrée** avec la **clef publique de l'émetteur**.
7. **L'empreinte** du message est calculée à l'aide de **la même fonction de hachage** que lors de la création de la signature.
8. L'empreinte obtenue en hachant le message en clair et l'empreinte obtenue en déchiffrant la signature sont comparées. Si elles sont identiques, **le message est authentique**.

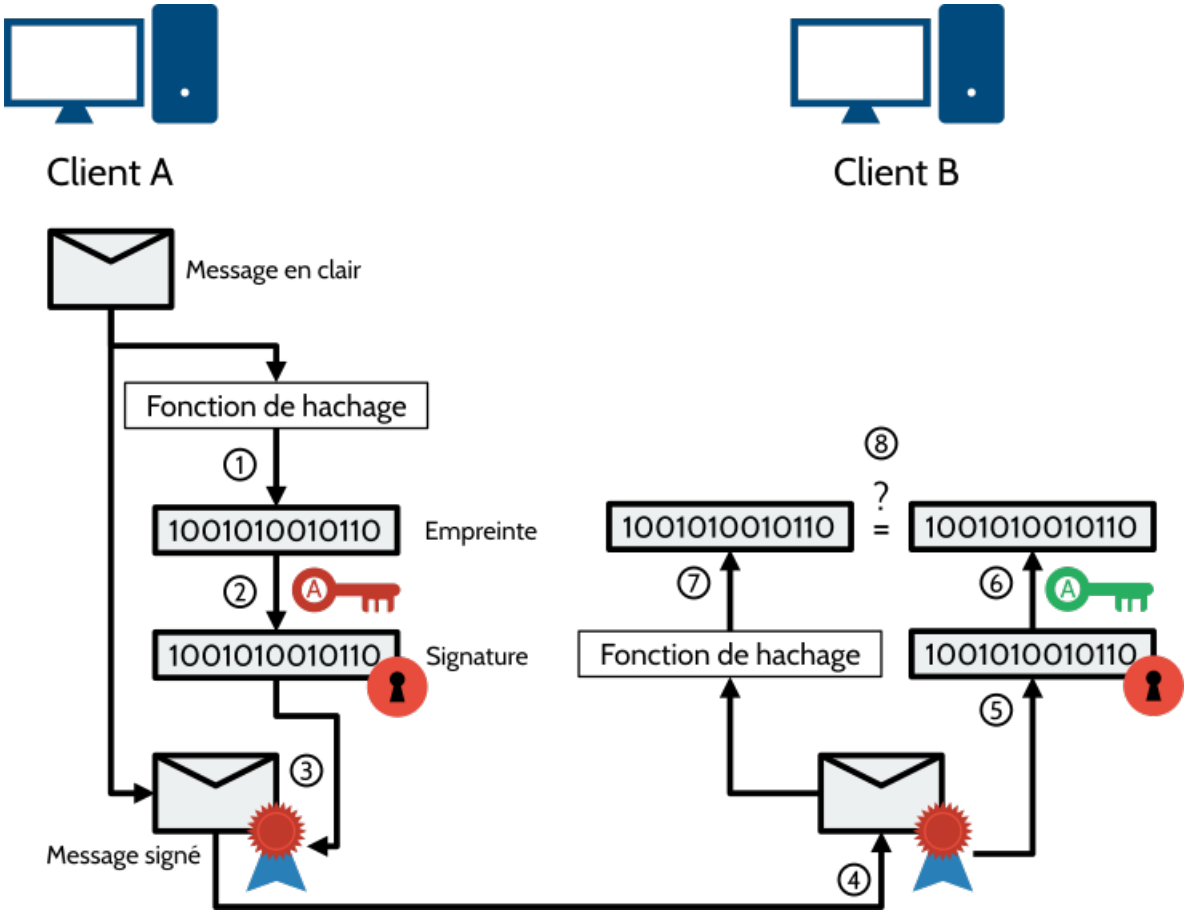


FIGURE 3 – Envoi d'un message signé

Exercice 2

Quelle clef est utilisée pour les opérations suivantes dans le cadre de l'envoi par Lucie d'un message à Simon :

1. Lucie chiffre son message avant de l'envoyer à Simon :
   ☐ Clef publique de L ☐ Clef privée de L ☐ Clef publique de S ☐ Clef privée de S
2. Lucie signe son message avant de l'envoyer à Simon :
   ☐ Clef publique de L ☐ Clef privée de L ☐ Clef publique de S ☐ Clef privée de S
3. Simon déchiffre un message chiffré reçu de Lucie :
   ☐ Clef publique de L ☐ Clef privée de L ☐ Clef publique de S ☐ Clef privée de S
4. Simon vérifie la signature d'un message signé par Lucie :
   ☐ Clef publique de L ☐ Clef privée de L ☐ Clef publique de S ☐ Clef privée de S

5 Les certificats

Les certificats sont l'équivalent d'une **carte d'identité numérique** pouvant être attribuée à des entités dont on souhaite vérifier l'identité (personnes, serveur, client, etc.).

Un certificat regroupe plusieurs éléments :

— La **clef publique** de l'entité à identifier

— Des **informations à propos du propriétaire du certificat** (nom, adresse courriel, localisation, etc.)

— Des **informations sur le certificat lui-même** (version, algorithme de signature, etc.)

— La **signature de l'autorité de certification** qui a produit le certificat. La signature est produite à partir des informations précédentes et avec **la clef privée de l'autorité de certification**.

La création d'un certificat s'effectue comme suit *(fig. 4)* :

1. Une demande de certificat (Certificate Signing Request en anglais, abrégé CSR) est produite. Elle regroupe les informations à propos de l'entité qui demande le certificat, ainsi que sa clef publique.
2. La demande de certificat est passée dans une fonction de hachage (par exemple SHA-256) pour obtenir une empreinte.
3. Cette empreinte est chiffrée à l'aide de la clef privée de l'autorité de certification. On obtient une signature.
4. La signature est ajoutée à la demande de certificat. On obtient ainsi un certificat signé.

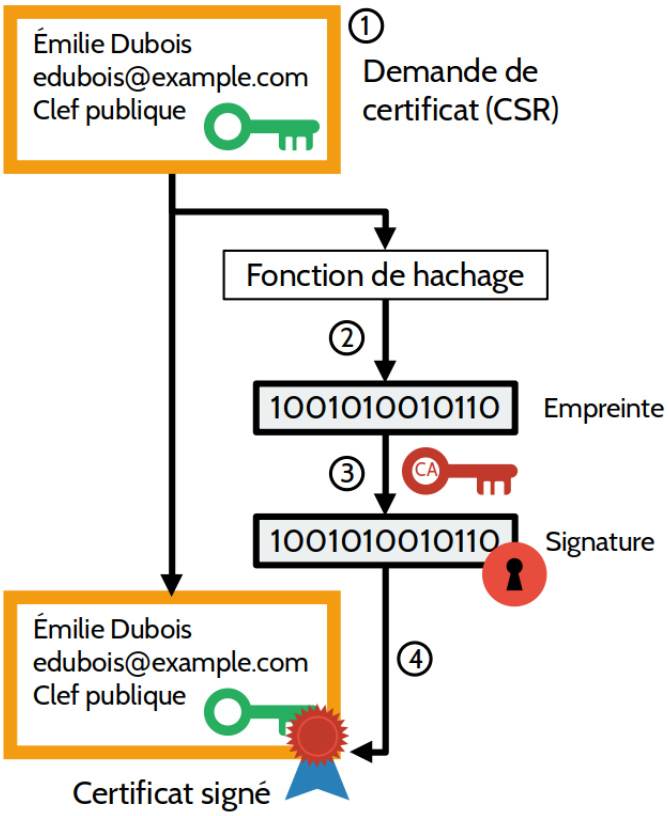


FIGURE 4 – Création d'un certificat

Voici le certificat de Wikipedia avec une partie des informations contenues *(fig. 5)*.

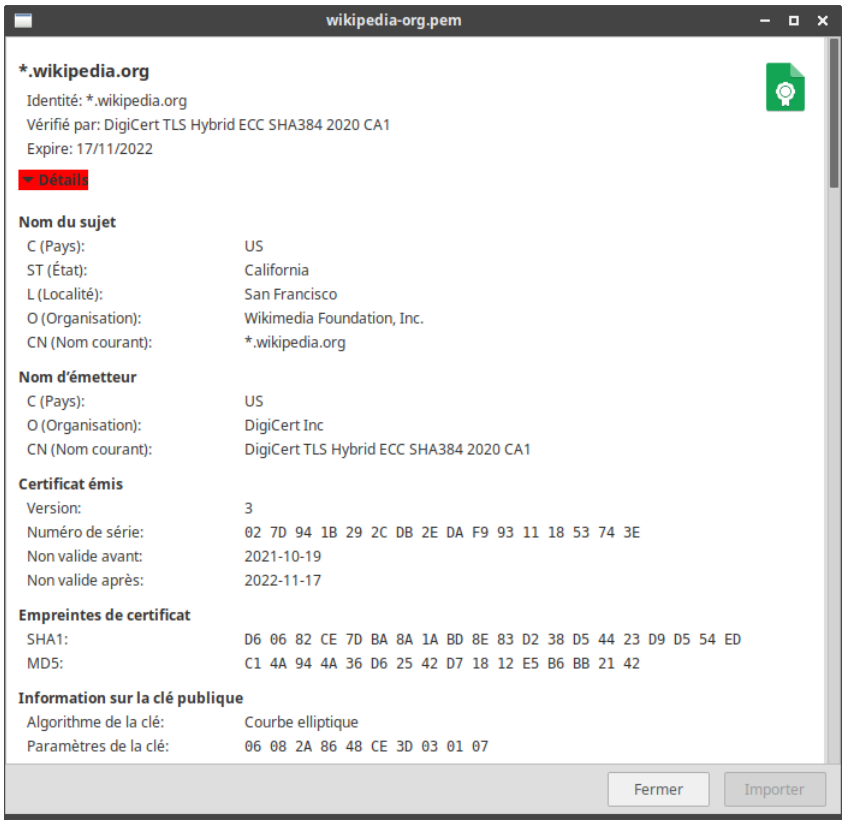


FIGURE 5 – Certificat de Wikipedia

Le certificat apporte plusieurs garanties :

Infalsifiable Le certificat étant signé, toute modification est impossible.

Certifié Il est approuvé par une autorité de certification faisant foi.

Nominatif Il est délivré à une et une seule entité.

Deux standards majeurs cohabitent afin de définir leur format et leur contenu :

X.509 Ce standard, créé en 1988 par l'Union Internationale des Télécommunications (UIT), est défini par la RFC 5280. Il s'appuie sur une **hiérarchie des autorités de certification** et n'admet qu'une seule signature d'autorité de certification.

OpenPGP Ce standard, créé en 1998 par l'IETF, est défini dans la RFC 2440. Il permet de signer un certificat par plusieurs autres certificats OpenPGP et ainsi construire une **toile de confiance**.

La vérification de la validité d'un certificat se fait en vérifiant les deux éléments suivants :

— Les informations portées par le certificat lui-même : période de validité, la signature du certificat, l'entité déclarée est bien celle qui vous a remis le certificat, etc.

— L'autorité qui a signé le certificat : vous faites confiance directement à l'autorité émettrice ou bien à l'autorité de niveau supérieur.

Un certificat auto-signé est un certificat signé par lui-même. C'est à vous de déterminer si oui ou non vous faites confiance au propriétaire du certificat.

Une autorité de certification qui émet un certificat possède également elle-même un certificat. Le certificat de l'autorité de certification peut être lui-même émis par une autre autorité de certification ou être auto-signé. En haut de la **chaîne de confiance**, on retrouve les **autorités racines**. On y retrouve :

— Les autorités de certification de votre entreprise, dans le cadre d'une infrastructure professionnelle.

— Des autorités publiques connues de tous telles que les services publics ou des autorités payantes (des sociétés privées dont le but est de susciter confiance et émettre, moyennant un paiement, des certificats).

Si un certificat ne peut pas être validé (lui-même et l'ensemble des autorités de la chaîne de confiance), l'utilisateur a le choix de lui faire tout de même confiance ou non.

Les certificats des autorités auxquels l'utilisateur fait confiance sont stockés dans un **magasin des certificats** où il est possible d'ajouter et de supprimer des certificats auxquels on fait confiance. Il est fourni par défaut avec une grande partie des autorités de certification publiques.

6 Les infrastructures à clefs publiques

Une **Infrastructure à clefs publiques** ou **Public Key Infrastructure (PKI)** en anglais désigne l'ensemble des composants, fonctions et procédures mis en place pour la création, la gestion ou la révocation des certificats à clefs publiques.

Le déploiement d'une PKI permet de centraliser et de simplifier la gestion des identités numériques, l'intégrité des documents et des échanges, l'assurance de la non répudiation.

Différentes composantes de la PKI remplissent des fonctions différentes :

Autorité d'enregistrement (Registry Authority abrégé en RA) Elle reçoit les demandes de certificats (Certificate Signing Request abrégé en CSR) de la part des utilisateurs, vérifie les informations communiquées par le demandeur et fournit une requête de certificat (un certificat non signé). Elle collecte également les demandes de révocation des certificats.

Autorité de certification (Certificate Authority abrégé en CA) Ce service reçoit les requêtes de certificats de l'autorité d'enregistrement et les signe. Par cette opération, l'autorité de certification transforme la requête en véritable certificat valide. Elle signe également les listes de révocation fournies par l'autorité d'enregistrement. **De par sa criticité, l'autorité de certification nécessite un haut niveau de sécurisation**.

Autorité de validation (Validation Authority abrégé en VA) Elle est en charge de recevoir les demandes de vérification de certificats et de répondre si un certificat est encore valide. Cette autorité est en charge notamment de vérifier si un certificat n'a pas été révoqué. Il est possible de télécharger localement la liste des certificats révoqués. Mais pour réduire le trafic réseau, le protocole OCSP est utilisé pour vérifier la validité d'un certificat auprès de l'autorité de validation.

Archivage La législation française impose un archivage des certificats (révoqués ou non), des clefs associées et des listes de révocation de certificats. Cette mesure vise a permettre l'exploitation de données à des fins judiciaires, même si le certificat est révoqué.

La création d'un certificat dans une PKI se fait comme suit *(fig. 6)* :

1. Le demandeur fournit les informations nécessaires à la création de son certificat à l'autorité d'enregistrement (en général au moyen d'un formulaire Web). Cette autorité va effectuer les vérifications nécessaires pour confirmer la validité des informations fournies.
2. Si les informations sont validées, l'autorité d'enregistrement va créer une demande de certificat et la communique à l'autorité de certification.
3. Selon le procédé vu précédemment, l'autorité de certification signe la demande de certificat et transmet le certificat au demandeur.
4. Le certificat est archivé pour des besoins légaux.

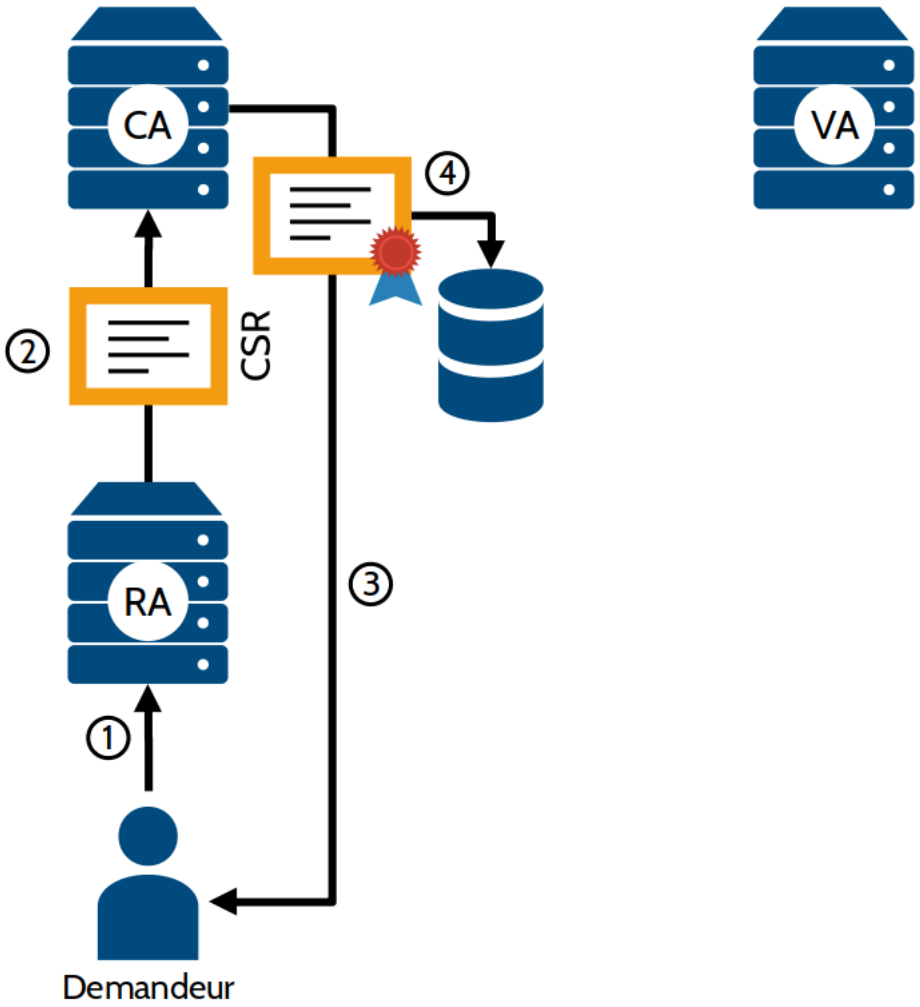


FIGURE 6 – Création d'un certificat au sein d'une PKI

La vérification des certificats par le client pose plusieurs problèmes :

— Le traitement par le client est long et nécessite beaucoup d'interrogations auprès des différentes parties de la chaîne de confiance.

— La propagation des listes de révocation auprès des clients prend du temps et est très coûteuse en trafic.

La centralisation de la tâche de vérification permet de palier à ces problèmes : une autorité de vérification se charge de faire les vérifications sur les certificats. Ce service, basé sur le protocole OCSP, se chargera de collecter les listes de révocation, de faire les contrôles sur les certificats et de répondre aux demandes des clients qui n'ont plus besoin de le faire.

La vérification de la validité d'un certificat dans le cadre d'une PKI se fait comme suit *(fig. 7)* :

1. L'autorité de validation récupère de manière régulière les listes des certificats révoqués (CRL) de la part de l'autorité de certification. Les listes sont également archivées à des fins légales.
2. Un expéditeur fournit son certificat à un destinataire (par exemple, un serveur Web à un client Web).

3. Le destinataire interroge l'autorité de validation afin de vérifier les informations contenues dans le certificat et si celui-ci n'est pas révoqué.
4. L'autorité de validation réponds si oui ou non le certificat est valide. Cette réponse est bien sûr chiffrée avec la clef privée de l'autorité de vérification afin d'éviter toute attaque.

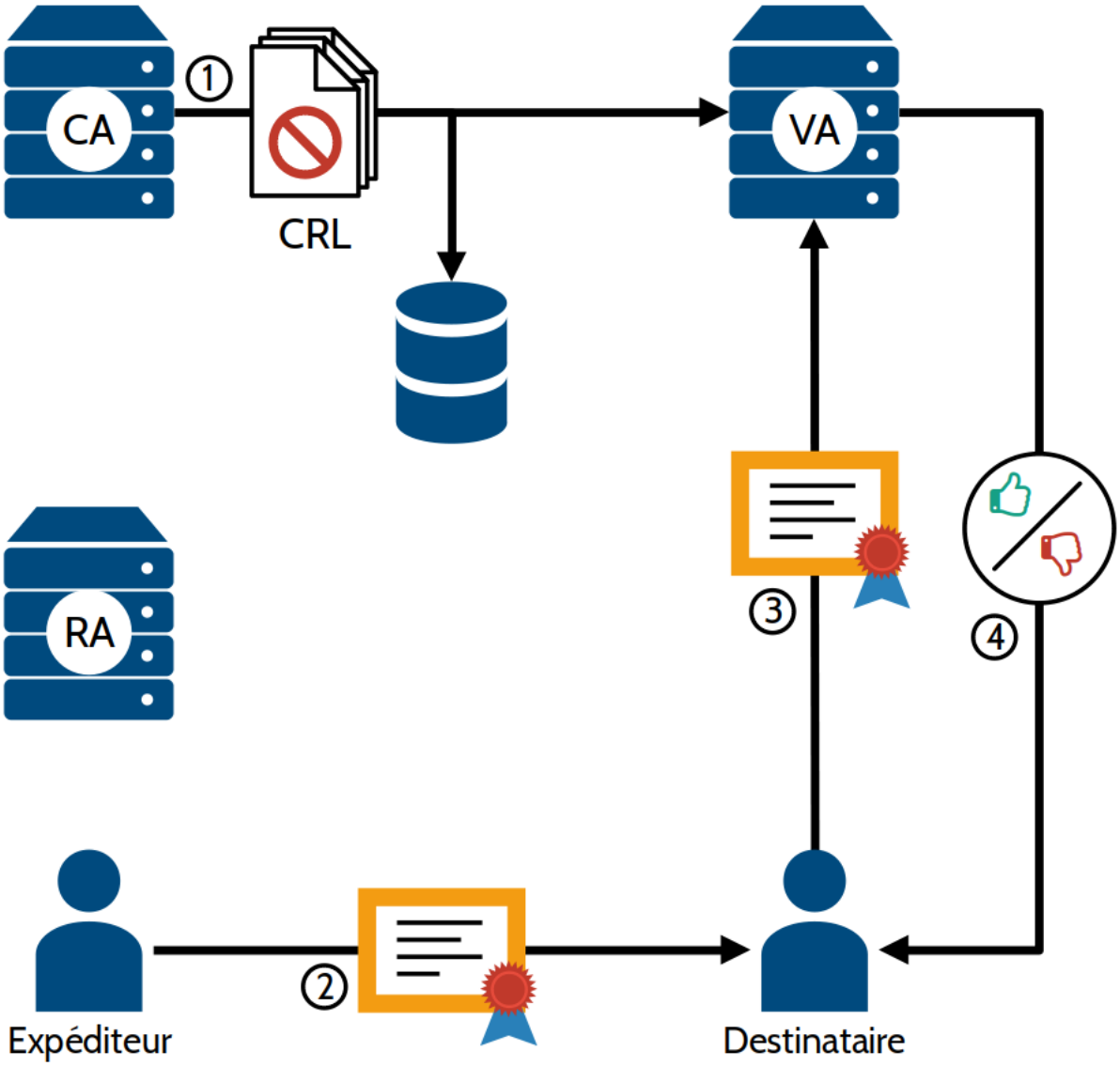


FIGURE 7 – Validation d'un certificat au sein d'une PKI

Voici une architecture recommandée dans le déploiement d'une PKI afin de garantir un maximum de sécurité :

— Une autorité de certification racine très protégée. Pour cela, le mieux est qu'elle soit hors ligne, mais son certificat auto-signé est publié.

— Cette autorité racine produit les certificats d'autorités de certification filles qui seront en ligne et qui signeront les demandes de certificats des utilisateurs.

— Si une autorité fille est corrompue, son certificat est révoqué. L'ensemble du système fonctionne toujours car l'autorité racine est préservée. Il suffit alors de produire une nouvelle autorité de certification fille pour continuer à fournir le service.

Exercice 3

Quelle autorité est en charge des opérations suivantes ?

1. Recueille les informations du demandeur et génère la CSR.
   ☐ CA ☐ RA ☐ VA ☐ Archivage
2. Signe la CSR pour former un certificat valide.
   ☐ CA ☐ RA ☐ VA ☐ Archivage
3. Émet la CRL.
   ☐ CA ☐ RA ☐ VA ☐ Archivage
4. Vérifie les informations contenues dans un certificat.
   ☐ CA ☐ RA ☐ VA ☐ Archivage
5. Stocke les certificats émis à des fins légales.
   ☐ CA ☐ RA ☐ VA ☐ Archivage

7 Le serveur proxy

Un serveur proxy ou serveur mandataire est un équipement placé entre un client et un serveur pouvant remplir plusieurs fonctions :

— Supervision et filtrage du trafic (censure, blocage des publicités, blocage des menaces, etc.)

— Mise en cache pour améliorer les performances

— Anonymisation des données

— Détection des menaces

— Etc.

De par sa position, un serveur proxy peut également être utilisé à des fins dangereuses. En effet, on peut placer un serveur proxy dans le cas d'un réseau WiFi ouvert afin de collecter des données personnelles, renvoyer de fausses informations, injecter des malwares, etc.

Un proxy transparent est un proxy ne nécessitant pas de paramétrage sur le client et qui n'est pas visible pour l'utilisateur.

L'interception du trafic chiffré est vue comme étant une attaque *man in the middle*. L'interception des flux chiffrés se fait en chiffrant/déchiffrant les flux au niveau du serveur proxy, ce dernier ayant sa propre autorité de certification afin de re-chiffrer la liaison entre le client et lui-même *(fig. 8)* :

1. Au préalable, le client doit ajouter le certificat du serveur proxy dans son magasin de certificats de confiance.
2. Le client envoie sa requête au serveur destinataire.
3. Le serveur proxy intercepte la requête, l'analyse et la transmet au serveur destinataire comme si elle provenait de lui.

4. Le serveur destinataire réponds au serveur proxy en fournissant sa clef publique.
5. Le serveur proxy déchiffre la communication et l'analyse.
6. Il chiffre la réponse avec un certificat généré à la volée pour le domaine et la transmet au client.
7. Le client déchiffre la réponse avec la clef publique du proxy.

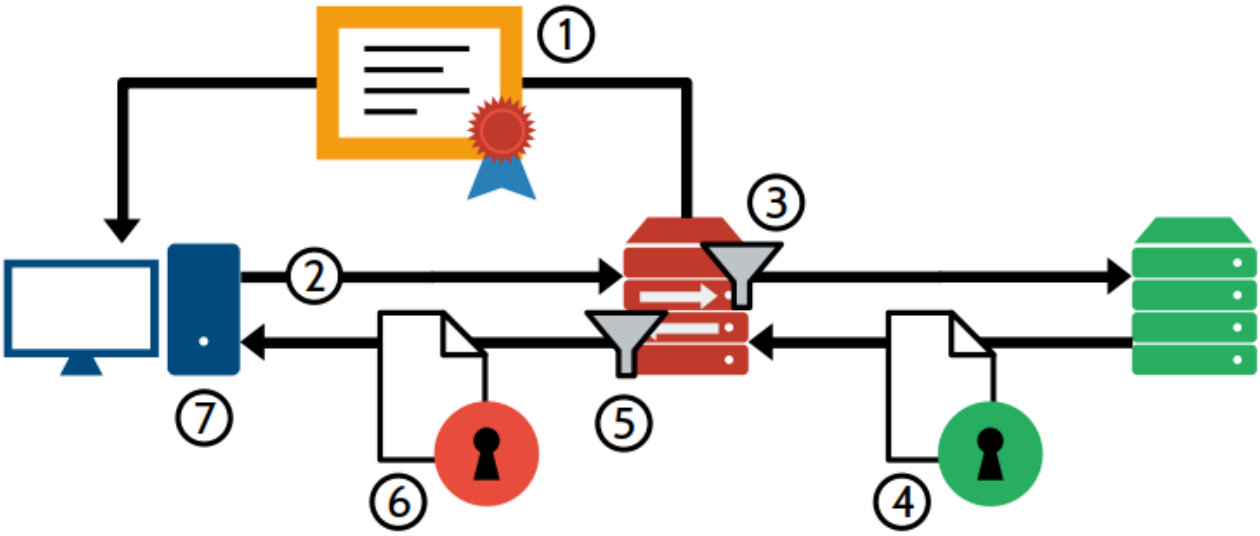


FIGURE 8 – Interception du trafic chiffré par un proxy

Un proxy inverse est un serveur proxy placé avant un service Web. Un utilisateur du service contactera le serveur proxy inverse pour accéder au service Web. Il permet d'effectuer les opérations suivantes :

— Mise en cache du contenu
— Équilibrage de charge entre plusieurs serveurs Web
— Assurer le chiffrement de la communication
— Gérer l'authentification du service Web

8 Le pare-feu et les ACL

Un pare-feu (ou *firewall* en anglais) est un dispositif matériel et/ou logiciel permettant de contrôler et filtrer les flux réseaux. Il permet notamment de filtrer le trafic entre le réseau interne et le réseau public. On parle alors d'une passerelle filtrante. On retrouve également des pare-feu à l'intersection des différentes zones réseau. Il protège des connexions et attaques malveillantes, il empêche la prolifération des attaques en cloisonnant le réseau.

Le pare-feu permet également d'auditer ou surveiller les flux entrants et sortants d'une zone car il s'agit du point de passage obligé pour passer d'une zone à l'autre.

Une *Access Control List* (ACL) est une liste de règles autorisant ou interdisant un trafic particulier sur un pare-feu, un routeur ou un serveur. Elles sont exécutées de haut en bas.

Il existe deux politiques possibles pour un pare-feu :

Liste noire Tout ce qui n'est pas interdit est autorisé. **Ce mode de fonctionnement est à bannir.**

Liste banche Tout ce qui n'est pas autorisé est interdit. **Ce mode de fonctionnement est à adopter.**

Chaque règle de pare-feu est composée des éléments suivants :

— Adresse IP source

— Adresse IP destination

— Type (TCP, UDP, ICMP, IP, etc.)

— Numéro de port

— L'action à effectuer (ACCEPT, REJECT ou DROP)

Il existe plusieurs types de pare-feu :

Pare-feu sans état Il s'agit de la méthode la plus simple et la plus ancienne de filtrage du trafic. Le pare-feu analyse les paquets individuellement, indépendamment les uns des autres. Il filtre selon les adresses IP source et destination, le port source et destination et le protocole réseau. Le fait d'analyser les paquets de manière individuelle ne permet pas de gérer la notion de session et est de moins en moins utilisé. Cela reste néanmoins un moyen simple de gérer quels services sont exposés.

Pare-feu à états Cette méthode suit la connexion entre le client et le serveur et tient compte des anciens paquets. On filtre selon l'adresse IP source et destination, le protocole et seulement le port destination à filtrer car le port source est choisi aléatoirement lors de l'établissement de la connexion.

Pare-feu applicatif Il s'agit du proxy, abordé plus en détail dans le chapitre dédié de ce cours. Il filtre le flux par application et vérifie que les requêtes sont bien conformes aux protocoles. Il s'agit du type de pare-feu permettant la plus grande finesse de règles et le plus consommateur en ressources. Ainsi, il convient de le placer en aval et sur une machine distincte d'un pare-feu dynamique pour limiter les requêtes à traiter en effectuant un premier tri. Ce type de pare-feu interprétant les requêtes transmises, il est vulnérables aux attaques sur la couche application.

Pour être efficace, **l'intégralité des flux entrants et sortants d'une zone doivent transiter par le pare-feu. Il doit être incontournable.** Il peut également constituer un point unique de défaillance : s'il vient à être hors service, la zone est soit coupée du reste du réseau, soit accessible sans aucune restriction.

8.1 Le pare-feu `nftables`

Le logiciel `nftables` permet le filtrage et la manipulation des paquets et des trames sous Linux. Il a pour vocation de remplacer les commandes `iptables`, `ip6tables`, `arptables` et `ebtables` et se base sur `netfilter`.

`nftables` peut intervenir à plusieurs étapes d'un flux réseau. Pour cela, on utilise des *hooks* :

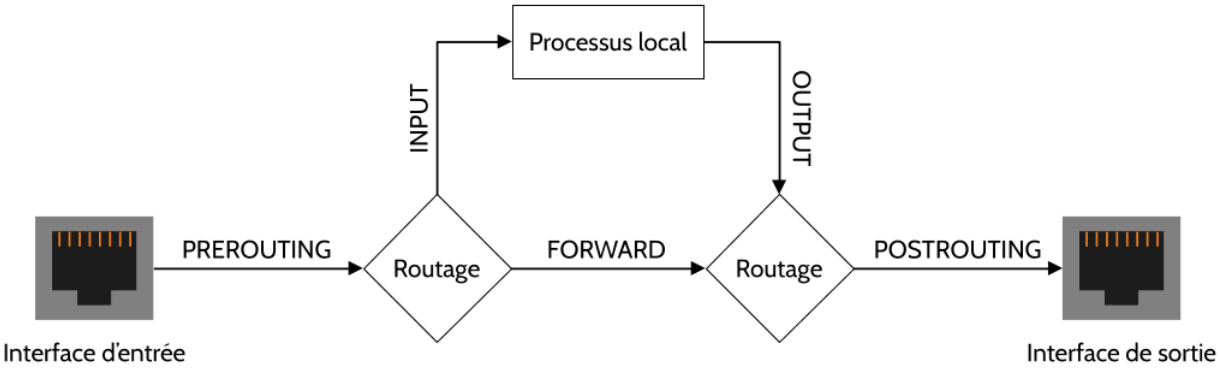


FIGURE 9 – Schéma simplifié de `nftables`

`PREROUTNG` Permet de traiter un paquet entrant avant son routage interne. Ce premier routage permet d'aiguiller un paquet s'il est destiné à notre hôte ou s'il doit être transféré ailleurs. On peut ici faire du NAT pour exposer un service d'une machine située en aval de notre hôte.

`INPUT` Traite les paquets ou trames destinés à notre hôte après le premier routage.

`FORWARD` Traite les paquets routés destinés à une machine en aval de notre hôte. Par défaut, Linux ne permet pas le routage entre les interfaces réseau. Il faut activer les options suivantes dans `sysctl` :

```
net.ipv4.ip_forward = 1           # Pour IPv4
net.ipv6.conf.all.forwarding = 1 # Pour IPv6
```

`OUTPUT` Pour le flux sortant des applications de notre hôte.

`POSTROUTING` Pour les paquets allant vers l'interface de sortie.

Les règles `nftables` sont stockées dans des **chaînes** qui décrit :

— le type de règles qu'elle contient :

`filter` Permet de filtrer les paquets IPv4 et IPv6 et trames ARP.

`nat` Permet de faire du NAT uniquement sur les paquets IPv4 et IPv6.

`route` Effectue du marquage de paquets, uniquement sur les paquets IPv4 et IPv6.

— son *hook*

— sa priorité d'application

— son action par défaut (ACCEPT, REJECT ou DROP). Sans spécification, l'action par défaut d'une chaîne est ACCEPT.

Les chaînes sont elles-mêmes contenues dans une **table** qui permet de regrouper les chaînes pour un type d'unité de données :

`ip` Agit sur les paquets IPv4 (table par défaut)

`ip6` Agit sur les paquets IPv6

`inet` Agit sur les paquets IPv4 et IPv6

`arp` Agit sur les trames couche liaison. Seuls les *hooks* `INPUT` et `OUTPUT` sont compatibles.

`bridge` Agit sur les interfaces type bridge.

`netdev` Agit en amont du routage interne. Les *hooks* utilisés sont `INGRESS` et `EGRESS`, que nous n'aborderons pas ici.

Il est possible de modifier à la volée les règles `nftables` mais celles-ci ne sont pas sauvegardées pour être rechargées au démarrage. Ainsi, nous préférerons configurer `nftables` depuis son fichier de configuration. Sous Debian et systèmes dérivés (Ubuntu, etc.), ce fichier se situe dans **/etc/nftables.conf**. Il faut redémarrer le service pour appliquer les changements.

Voici un exemple de fichier de configuration `nftables` :

```
1  #!/usr/sbin/nft -f
2  flush ruleset
3  table inet tableinet {
4     chain input {
5        type filter hook input priority filter; policy drop;
6        iifname lo accept
7        icmp type echo-request accept
8        tcp dport 22 accept
9        ct state {established,related} accept
10    }
11    chain postrouting {
12       type nat hook postrouting priority srcnat;
13       iifname enp1s0 masquerade
14    }
15 }
16 table ip tableip {
17    chain prerouting {
18       type nat hook prerouting priority dstnat;
19       tcp dport 2222 dnat to 10.50.0.2:22
20    }
21 }
```

Décrivons cette configuration :

— Une première table nommée `tableinet` gérant les paquets IPv4 et IPv6 (lignes 3 à 24).

  — Une première chaîne `filter INPUT` avec comme politique par défaut DROP (lignes 4 à 10).

    — Le flux entrant depuis l'interface `lo` est accepté (ligne 6).

    — Les requêtes ICMP PING sont acceptées (ligne 7).

    — Le flux sur le port TCP 22 est autorisé (SSH) (ligne 8).

        - Le trafic déjà établi est autorisé (ligne 9).
    - Une chaîne `nat POSTROUTING` (lignes 20 à 23).
        - Le flux entrant depuis l'interface `enp1s0` sera natté (ligne 22).
- Une seconde table nommée `tableip` gérant uniquement les paquets IPv4 (lignes 25 à 30).
    - Une chaîne `nat PREROUTING` (lignes 17 à 20).
        - Le flux entrant sur le port TCP 2222 sera transféré vers le port TCP 22 de l'hôte `10.50.0.2` (ligne 19).

Un memento regroupant les règles usuelles est disponible à la fin de ce fascicule.

Exercice 4

Pour le serveur suivant dont l'adresse IP est 10.50.0.1/24, écrivez les ACL dans l'ordre :

- Service Web écoutant sur le port TCP 80.
- Service Web écoutant sur le port TCP 443.
- Service SSH écoutant sur le port TCP 2223 exposé uniquement sur le réseau interne.
- Service DNS écoutant sur le port UDP 53 exposé uniquement sur le réseau interne.
- Ping (ICMP) autorisé uniquement sur le réseau interne.
- Tout le reste du trafic est bloqué.

Adresse source	**Adresse destination**	**Protocole**	**Port**	**Action**
Exemple : 0.0.0.0/0	*10.50.0.1/24*	*UDP*	*67*	*ACCEPT*

9 Le VPN

Ce cours n'est sponsorisé par aucun service de VPN commercial, contrairement aux vidéos YouTube.

Le réseau privé virtuel, en anglais *Virtual Private Network* (VPN), est un réseau créé virtuellement en utilisant le réseau Internet. Il permet de créer un lien direct entre deux machines distantes et isole leurs communications entre-elles du reste du réseau public.

Le VPN est une solution à moindre frais de raccorder de manière sécurisée différents sites d'une entreprise, une entreprise à un partenaire ou de permettre aux employés d'avoir recours au télétravail ou au travail nomade. Les données transitent de manière chiffrée dans un **tunnel VPN**.

Un VPN permet de se prémunir des attaques suivantes :

Écoute et modification du trafic La communication étant chiffrée, la confidentialité des communications est assurée.

Usurpation d'identité L'authentification mutuelle entre les deux réseaux empêche les attaques du type *man in the middle*.

Rejeu de la communication Il est impossible de rejouer une session.

Attaques par rebond Le tunnel est ouvert uniquement entre les réseaux autorisés.

Il existe différent protocoles VPN dont voici les plus courants :

L2TP Protocole de couche 2 normalisé par les RFC 2661 et RFC 3931 permettant de transporter des sessions PPP, notamment entre les opérateurs de collecte de trafic et les FAI.

IPsec Protocole de couche 3, issu des travaux de l'IETF. Il permet de transporter des données chiffrées pour les réseaux IP. Il s'agit du protocole le plus utilisé.

MPLS Il est possible de créer des VPN distribués sur un coeur de réseau MPLS pour les couches 2 et 3. Cette solution est utilisée par les opérateurs pour fournir aux clients professionnels des solutions de liaisons entre leurs différents sites.

SSL/TLS Cela désigne à la fois les VPN en client léger (sur navigateur Web, ne nécessitant pas d'installation de logiciel spécifique), ainsi que des technologies de VPN telles que OpenVPN.

Concernant les services de VPN commerciaux, il est important de prendre en compte que la communication est chiffrée entre le client et l'extrémité du tunnel VPN. Une fois à l'extérieur du tunnel, la communication peux-être de nouveau vulnérable aux attaques citées ci-dessus. Le fournisseur de service VPN accède aux communications en clair car il se charge du déchiffrement. Ainsi, il est primordial de s'assurer de l'usage fait par le fournisseur de VPN des données qui transitent.

10 Architectures réseau sécurisées et DMZ

Les règles d'architecture réseau décrites ici sont dérivées du guide "Recommandations relatives à l'interconnexion d'un système d'information à Internet", disponible sur le site de l'ANSSI : https://www.ssi.gouv.fr/uploads/2020/06/anssi-guide-passerelle_internet_securisee-v3.pdf

Afin d'isoler et appliquer les politiques adaptées aux différents usages (bureautique, serveurs à usages internes, serveurs à usages externes, etc.), il est nécessaire de segmenter le réseau en plusieurs zones distinctes séparées par des routeurs et pare-feu assurant la jonction et le filtrage des flux interzones.

Une première règle à respecter dans le déploiement d'un réseau d'entreprise est de séparer le SI interne du reste d'Internet par une zone démilitarisée (DMZ). Cette zone, faisant office de tampon assurant un filtrage efficace des flux, est connectée de part et d'autre par des routeurs et des pare-feu

distincts et maîtrisés, c'est-à-dire gérés intégralement par votre organisation. Les services exposés sur Internet seront situés dans cette DMZ et sur une infrastructure physique dédiée. Il est vivement recommandé que ces routeurs et pare-feu soient sur des infrastructures physiques différentes. La DMZ ne doit pas contenir de services essentiels pour le fonctionnement de la société et doit être considérée comme perdable. Il faut également que les différents équipements de sécurité (pare-feu, proxy, routeur, annuaire, etc.) soient sur des équipements physiques distincts *(fig. 10)*.

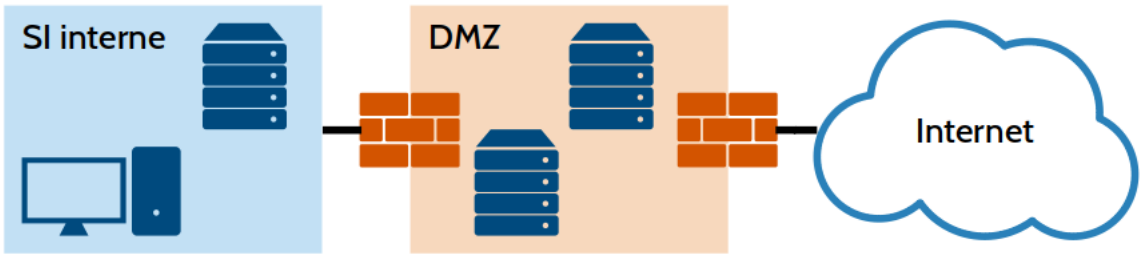


FIGURE 10 – Mise en place d'une DMZ entre le réseau interne et Internet

Les flux entre le SI interne et Internet doivent également transiter par un serveur proxy situé en DMZ. Celui-ci doit être incontournable, en mode explicite, configuré en local sur chaque poste client et sera en charge du filtrage du trafic et de la journalisation des activités. Votre serveur proxy se chargera de la rupture de flux (agir en intermédiaire entre le client et le serveur finaux d'une communication en empêchant un lien direct entre ceux-ci) et appliquera les règles de sécurité de votre entreprise *(fig. 11)*.

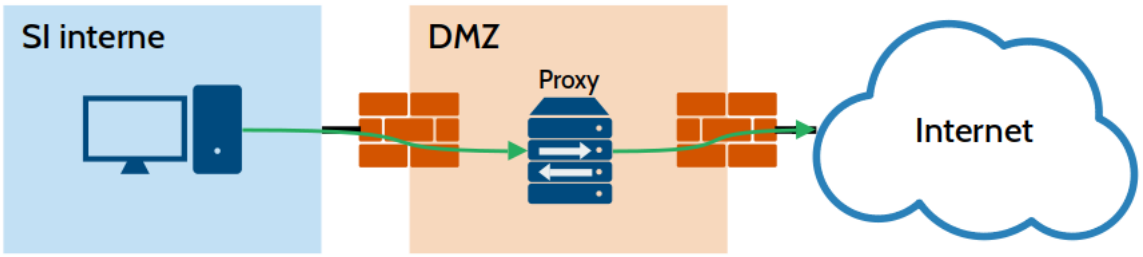


FIGURE 11 – Le trafic depuis le SI interne passe par le proxy

Il est crucial pour la sécurité de votre SI que les éléments situés en DMZ ne puissent pas requêter directement l'annuaire situé dans le SI interne. Vous pouvez pour résoudre cela :

— Créer une copie minimaliste et en lecture seule de l'annuaire interne *(fig. 12)*.

— Créer un serveur proxy entre vos services et l'annuaire

— Créer un serveur proxy en charge de l'authentification au sein de votre SI interne pour les services en DMZ nécessitant une authentification

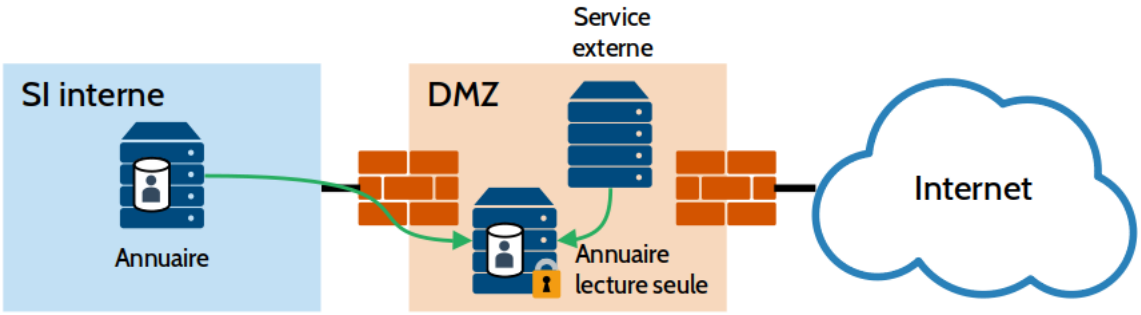


FIGURE 12 – Création d'une copie lecture seule de l'annuaire

Vous pouvez découper votre DMZ en plusieurs zones remplissant différentes fonctions *(fig. 13)*. On peut retrouver :

1. La zone d'accès interne avec le serveur proxy et le filtrage depuis et vers la zone interne
2. La zone des services internes avec le résolveur DNS (voir plus loin) et la copie d'annuaire pour le proxy
3. La zone des services exposés sur Internet
4. La zone des services relais comporte la rupture des flux, l'analyse du trafic depuis et vers Internet, les terminaisons VPN, les proxy Web et le relais de messagerie (voir plus loin).
5. La zone d'accès externes en charge du filtrage du trafic

Il est recommandé de séparer ces différentes zones par des pare-feu et d'utiliser des commutateurs dédiés au sein de chaque zone (ou à défaut, des VLAN différents).

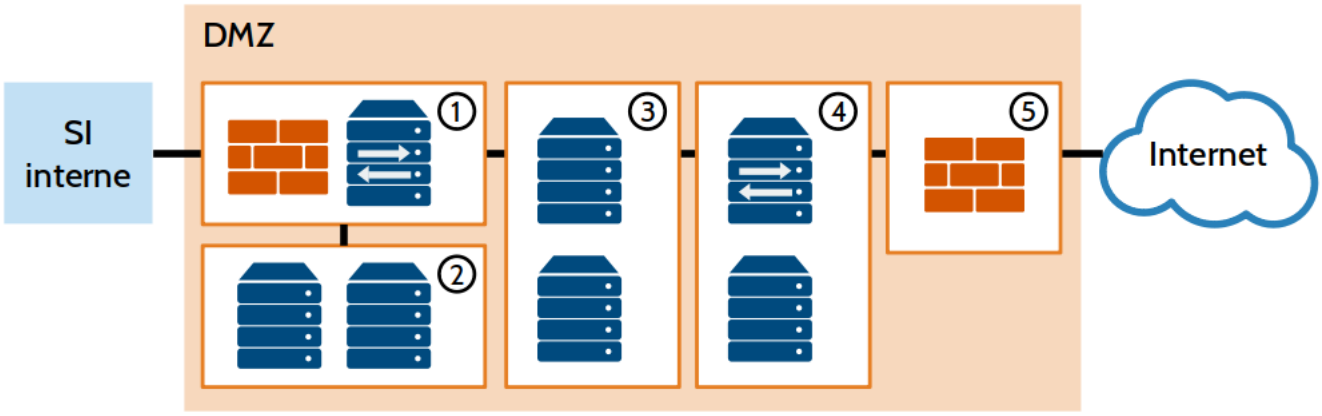


FIGURE 13 – Découpage de la DMZ en plusieurs zones

Les clients internes du SI ne doivent pas être en mesure de requêter de serveur DNS autre que celui du SI interne dédié aux domaines locaux. Seul le serveur proxy doit être en mesure de requêter un résolveur DNS placé dans la DMZ *(fig. 14)*.

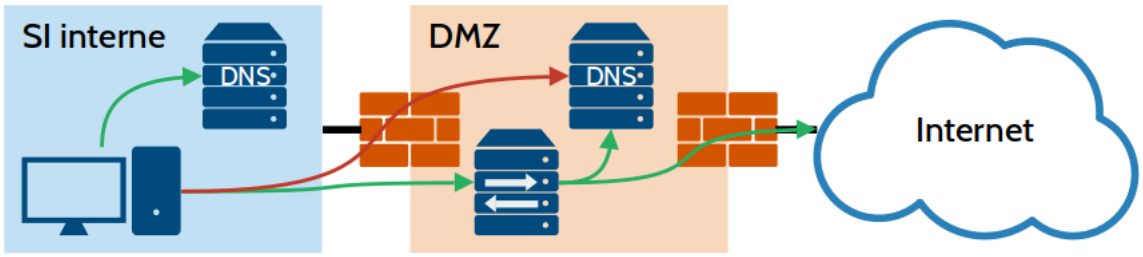


FIGURE 14 – Détail des requêtes DNS

Dans le cas d'une organisation comprenant plusieurs sites, chaque site doit disposer de sa DMZ propre. L'interconnexion VPN entre les différents sites, un cloud privé ou tout autre service étant interconnecté *via* Internet ne doit pas aboutir dans le SI interne, mais dans la DMZ comme un flux provenant de l'extérieur, dans la zone des services relais. Si les différents sites accèdent à Internet, s'assurer que la politique de sécurité est homogène *(fig. 15)*.

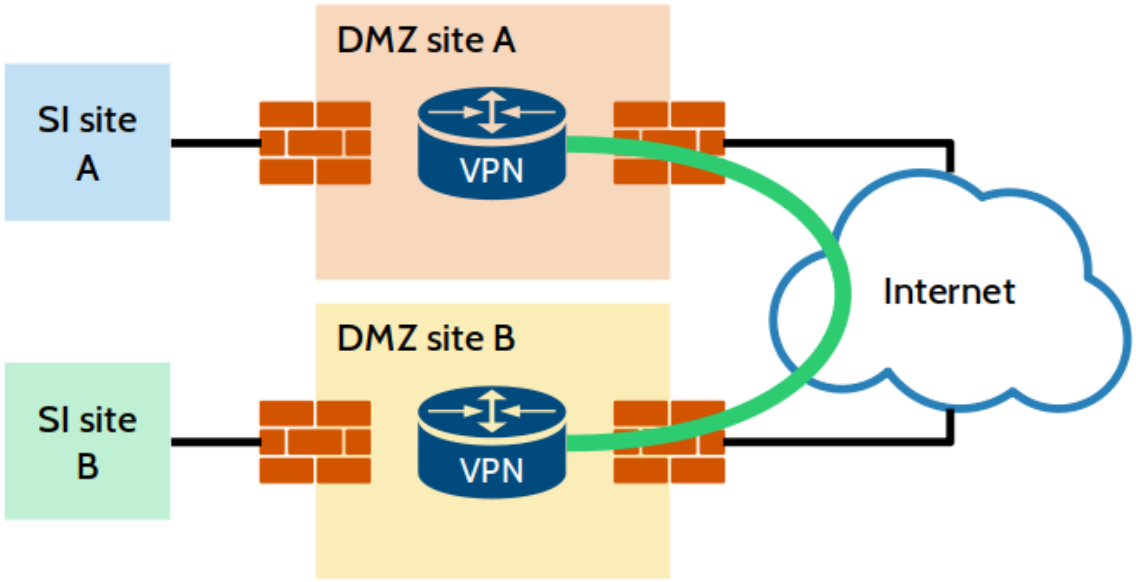


FIGURE 15 – Mise en œuvre d'un VPN multisite

Les routeurs, hors du routeur d'interconnexion vers Internet, doivent être configurés statiquement et ne doivent pas permettre l'usage des protocoles de routage dynamique (RIP, OSPF, etc.). Les pare-feu doivent être configurés pour ignorer les paquets et non les refuser. Configurez l'ensemble des services afin qu'ils communiquent le moins d'informations possibles sur votre infrastructure. Chaque personne se voit attribuer un compte nominatif et personnel (pas de comptes génériques). Centralisez les journaux de vos équipements dans un serveur unique renforcé.

Pour les services Web, des proxy inverses doivent être mis en place pour assurer l'authentification des accès, le chiffrement de la communication, le contrôle d'accès, la journalisation, etc. *(fig. 16)*

FIGURE 16 – Mise en œuvre d'un proxy applicatif

Pour les services de messagerie, le Mail Transfer Agent (MTA, serveur d'envoi) et le Mail Delivery Agent (MDA, serveur stockant les boîtes de messagerie) doivent être séparés. Le MTA doit être placé dans la zone des services relais alors que le MDA doit être situé dans le SI interne. Le MTA doit analyser et filtrer les messages entrant et sortants (spam, virus, etc.). Éviter le déploiement d'un service Webmail qui engendrerait un flux jusque dans le SI interne *(fig. 17)*. Seuls les protocoles chiffrés doivent être utilisables par les clients (SMTPS, IMAPS et POPS). Comme vu pendant le cours de services réseaux avancés, déployez le SPF, DKIM, DMARC et DNSSEC.

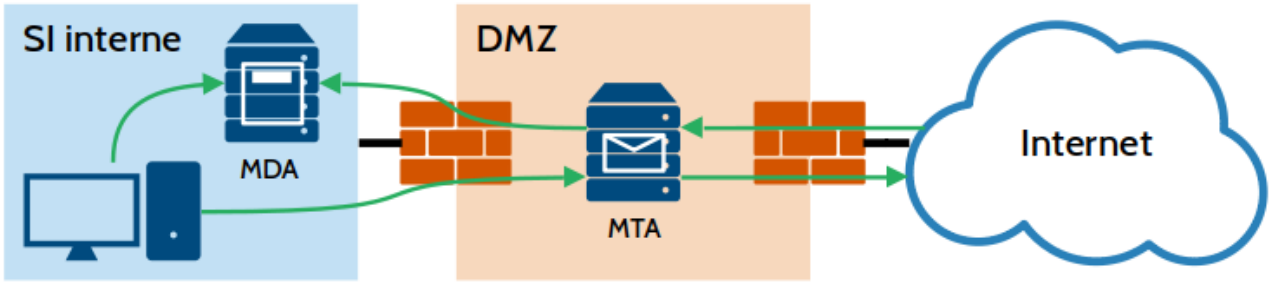


FIGURE 17 – Architecture d'un service mail avec une DMZ

Exercice 5

Concevez une architecture sécurisée intégrant les services suivants et tracez les flux autorisés :

— Annuaire LDAP

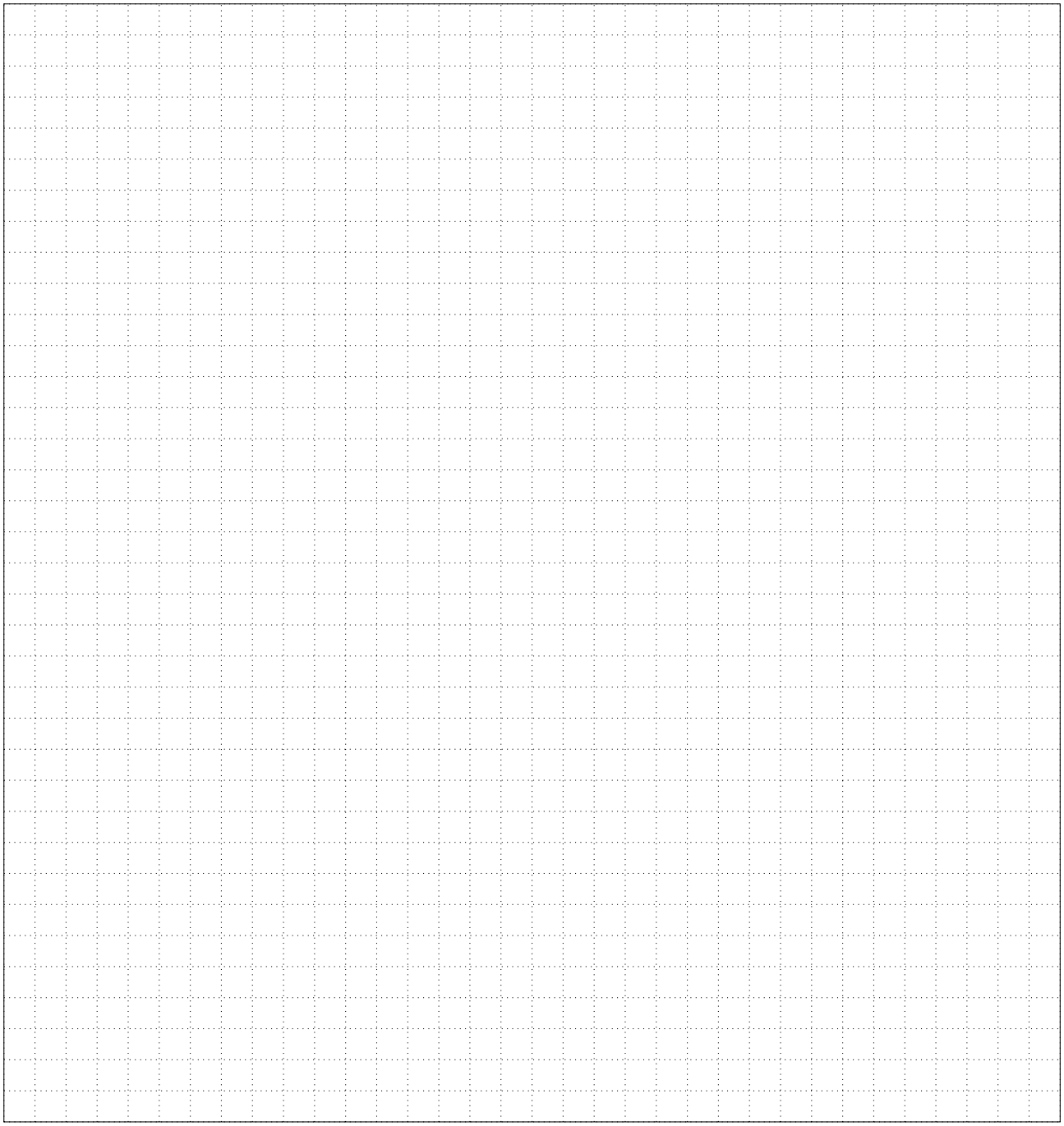
— Serveur Web destiné à l'usage interne uniquement

— Serveur Web exposé sur Internet

— Serveur proxy authentifié avec LDAP

— Service de messagerie (serveur IMAP, SMTP et webmail authentifié avec LDAP)

— Serveur VPN authentifié avec LDAP



11 Sécurisation des services réseaux

Au-delà des durcissements à réaliser sur votre réseau, ainsi que sur le système d'exploitation, des opérations sont à effectuer sur les services s'exécutant sur vos serveurs :

Effectuer régulièrement les mises à jour Les mises à jour publiées par les différents éditeurs corrigent des failles et apportent des améliorations de sécurité. Il est primordial d'appliquer régulièrement les mises à jours de système d'exploitation, ainsi que des logiciels installés sur le serveur.

Supprimer ou désactiver tout ce qui est inutile Afin de réduire la surface d'attaque, il est nécessaire de désactiver et supprimer tout ce qui n'est pas strictement nécessaire au bon fonctionnement du serveur. Tout service, fichier, application superflu offrent une surface d'attaque potentielle à un attaquant. Par exemple, une interface graphique sur un serveur Linux est superflu. N'activer que les modules de vos services qui sont nécessaires.

Changer le port d'écoute quand cela est possible À part pour les services exposés aux utilisateurs (internes ou externes), il est recommandé de changer le port d'écoute par défaut des services. Cela est notamment vrai pour SSH, qui est particulièrement ciblé par les attaquants.

Limiter les interfaces et adresses d'écoute Lorsqu'un service n'est utilisé que par un réseau spécifique, comme le réseau interne ou en rebouclage du serveur, il est recommandé de limiter l'écoute des services aux adresses IP autorisées et aux interfaces sur lesquelles les requêtes légitimes arrivent.

Installer fail2ban Cet outil analyse les journaux des services et bloque les attaques en force brute. Il est rapide à installer, facile à configurer et apporte une protection supplémentaire.

Ne pas exécuter les services en root En cas de corruption d'un service, il convient de limiter le champ d'action de l'attaque en exécutant les services réseaux avec des comptes techniques n'ayant accès qu'à ce qui est nécessaire pour fonctionner. L'usage du compte `root` pour l'exécution des services réseaux est à proscrire.

Isoler les services sur des machines séparées Pour limiter la propagation d'une attaque, il convient d'isoler les services réseaux sur des machines (virtuelles ou physiques). À défaut, l'usage de conteneurs permet d'apporter une meilleure isolation que d'exécuter directement tous les services sur une machine unique.

Forcer le chiffrement des communications Pour sécuriser les communications entre les clients et le serveur, il est primordial d'activer le chiffrement des communications et de désactiver la version non chiffrée. Pour HTTP, il est possible de paramétrer le serveur Web pour rediriger le trafic vers HTTPS. Pour Apache, cela se configure comme suit :

```
RewriteEngine On
RewriteCond %{SERVER_NAME} =<votre domaine>
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=
    permanent]
```

Appliquer une politique de moindre habilitation Chaque utilisateur peut être, volontairement ou non, source d'une attaque (connexion à partir d'un poste infecté, erreurs, acte malveillant).

Ainsi, il convient d'habiliter les utilisateurs uniquement aux services auxquels ils ont besoin. Il ne faut pas confondre besoin et envie : un utilisateur peut dire avoir besoin d'un service, il faut alors vérifier la nécessité de l'habilitation demandée avant de la lui accorder. Une fois qu'un utilisateur n'a plus besoin des habilitations (mutation, départ, etc.), il est primordial de les lui retirer.

Journaliser les événements sur un serveur de journalisation Lors de la compromission d'un service, ce dernier conservera trace dans les journaux. Ainsi, il convient d'activer la journalisation des services et faire une copie des journaux vers un serveur de journalisation dédié et correctement sécurisé.

Sécuriser l'accès physique aux serveurs Lorsqu'un attaquant a un accès physique au serveur, il est très aisé pour lui d'effectuer une action malveillante dessus. Il convient alors d'isoler physiquement les serveurs dans un local dédié, fermé et sécurisé et dont les accès sont attribués aux seules personnes ayant nécessité de pénétrer dans le local. Un serveur ne doit en aucun cas être mutualisé avec un poste utilisateur.

Mettre en place une supervision des services Il est nécessaire de surveiller le bon fonctionnement des services réseaux, ainsi que la bonne santé des serveurs les exécutant. Pour cela, une solution de supervision, qui indiquera en temps réel toute anomalie afin de réagir le plus rapidement possible, sera mise en place.

12 Pour aller plus loin

De nombreux guides, fournis par l'Agence Nationale de la Sécurité des Systèmes d'Information, expliquent plus en détails comment installer et configurer les différents composants d'un réseau d'entreprise de manière sécurisée.

Les guides sont disponibles ici : `https://www.ssi.gouv.fr/entreprise/bonnes-pratiques`

Modalités d'examen final

L'examen final sera effectué sur ordinateur. Voici sa composition :

— Questions de cours : 5 points

— Exercices pratiques : 15 points

Les modalités d'examen sont les suivantes :

— Temps imparti : 2h

— Documents papier autorisés.

— Calculatrice interdite.

— Ordinateurs personnels, montres connectées et téléphones portables interdits.

Cet examen vise à vérifier vos capacités propres d'analyse et d'autonomie. Ainsi, la copie de la configuration produite par un autre étudiant ou l'usage de l'intelligence artificielle sont fortement déconseillés car ils ne vous permettront pas de développer efficacement les savoir-faire nécessaires pour l'examen et votre future vie professionnelle.

Munissez-vous le jour de l'examen de votre adresse courriel IUT. Un compte sera créé sur la plateforme d'examen. Le mot de passe vous sera communiqué à l'aide d'un coupon situé à votre place.

L'examen sera effectué en salle informatique coupé d'Internet. Ainsi, il est important pour vous de prévoir en amont l'ensemble des documents nécessaires à l'examen.

Chaque examen est individuel et généré aléatoirement afin de prévenir toute tentative de triche. **Toute tentative de triche sera punie par une sanction disciplinaire.**

Les questions de cours porteront sur des éléments du cours et sanctionneront également votre esprit d'analyse face à des symptômes. **Une fois les questions de cours validées, vous ne pourrez pas modifier vos réponses.**

Les exercices pratiques sont similaires aux travaux pratiques que vous aurez réalisés en séance. Il vous sera demandé d'effectuer des opérations sur une infrastructure existante. **C'est pourquoi il est très vivement recommandé de produire une documentation papier de vos travaux pratiques à apporter le jour de l'examen.**

La correction de l'examen sera effectuée par des tests automatiques. Cependant, contrairement aux travaux pratiques, vous ne pourrez pas exécutez vous-même le robot de tests automatiques. Vous devrez vous assurer du bon fonctionnement de votre solution manuellement.

Travaux pratiques

Les travaux pratiques, présentés ici, sont des mises en situation analogue aux demandes qui seront susceptibles de vous être proposées lors de votre vie professionnelle.

Ce sujet se déroule sur l'ensemble des séances de TP pendant lesquelles vous serez mis en autonomie.

Les travaux pratiques seront corrigés à l'aide de tests automatiques afin de vérifier le bon fonctionnement de votre solution.

Vous retrouverez le jour de l'examen une mise en pratique similaire à ce que vous ferez en TP.

Il est très vivement recommandé de produire une documentation papier de vos travaux pratiques à apporter le jour de l'examen.

Sujet 1 : Pare-feu et serveur proxy

La société Pelletier Éditions est spécialisée dans l'impression et la distribution de livres. Elle souhaite sécuriser l'accès à Internet aux usagers de son SI interne. Pour cela, elle fait appel à vous afin de mettre en œuvre un serveur proxy par lequel le trafic sortant du SI interne sera forcé de transiter. Le domaine est `pelletiereditions.fr`

Votre machine virtuelle possède deux interfaces :

— `eth0` : Interface connectée à la DMZ et à Internet. L'adresse est obtenue *via* DHCP dans la plage d'adresse 10.50.0.0/16.

— `eth1` : Interface connectée au SI interne. L'adresse est obtenue *via* DHCP dans la plage d'adresse 10.60.0.0/16.

L'authentification des utilisateurs du proxy se fera depuis le serveur LDAP de l'entreprise. Celui-ci est hébergé sur 10.50.255.254. Les utilisateurs sont stockés dans `ou=Utilisateurs,dc=pelletiereditions,dc=fr`. Le compte utilisé pour interroger l'annuaire est `cn=proxy,dc=pelletiereditions,dc=fr` dont le mot de passe est `tprezo`.

Seuls les domaines suivants seront autorisés par le proxy :

— deb.debian.org

— security.debian.org

— fr.wikipedia.org

— intranet.pelletiereditions.fr

Les flux suivants doivent être paramétrés dans le pare-feu nftables :

Adresse source	**Port source**	**Adresse dest.**	**Port dest.**	**Description**
Tous	ICMP	Interface `eth0`	ICMP	Ping `eth0`
Tous	ICMP	Interface `eth1`	ICMP	Ping `eth1`
Tous	UDP 67	10.50.255.254	UDP 67	Trafic DHCP
10.60.0.0/16	TCP 80	Proxy	TCP 3129	Trafic HTTP SI interne envoyé dans votre proxy
10.60.0.0/16	TCP 443	Proxy	TCP 3130	Trafic HTTPS SI interne envoyé dans votre proxy
Proxy	Tous	Tous	TCP 80	Trafic sortant HTTP du proxy
Proxy	Tous	Tous	TCP 443	Trafic sortant HTTPS du proxy
Tous	TCP 8081	10.60.255.254	TCP 8089	Serveur Web SI interne

Sujet 2 : Le VPN

L'entreprise Iris Architectes est un bureau d'architecture en bâtiment basé à Beaune. Pour offrir une meilleure qualité de vie à ses salariés, elle veut mettre en place la possibilité de télétravailler. La solution de VPN OpenVPN a été retenue.

L'authentification des utilisateurs du proxy se fera depuis le serveur LDAP de l'entreprise. Celui-ci est hébergé sur 10.50.255.254. Les utilisateurs sont stockés dans `ou=Utilisateurs,dc=irisarchi,dc=com`. Le compte utilisé pour interroger l'annuaire est `cn=vpn,dc=irisarchi,dc=com` dont le mot de passe est `tprezo`. Les clefs TLS devront être générées aléatoirement et communiquées au client.

Le VPN doit permettre l'accès au réseau 10.50.0.0/16 et doit écouter sur le port UDP 1194. Le port TCP 80 doit être ouvert sur le pare-feu.

Pour faciliter la distribution de la configuration OVPN aux utilisateurs, une API devra être développée sur votre serveur. Cette API recevra en paramètre GET le nom d'utilisateur dans le champ `user`. La réponse devra être un fichier JSON contenant les éléments suivants :

Clef JSON	**Valeur attendue**
`login`	Identifiant utilisateur.
`tlskey`	La clef TLS générée par le serveur.
`ovpn`	Le contenu du fichier OVPN pour l'utilisateur demandé.

Appel d'API avec un utilisateur existant dans le LDAP :

```
GET /?user=pdubois

HTTP/1.0 201 Created
Content-Type: application/json
{"login":"pdubois","tlskey":"<mot de passe de la clef TLS>","ovpn
   ":"<contenu du fichier OVPN>"}
```

Appel d'API avec un utilisateur n'existant pas dans le LDAP :

```
GET /?user=invaliduser

HTTP/1.0 404 Not Found
```

Si l'API est rappelée pour un utilisateur pour lequel des clefs ont déjà été générées, les anciennes clefs doivent être détruites et de nouvelles clefs doivent être régénérées.

Sujet 3 : L'infrastructure de clefs publiques

La société Papin Énergies est une multinationale de production d'électricité, de biogaz et d'hydrogène. Son infrastructure informatique est très grande et elle souhaite centraliser et automatiser la distribution des certificats TLS utilisés dans ses serveurs internes. Vous devrez mettre en place une infrastructure à clefs publiques. Les certificats devront utiliser le chiffrement RSA 2048 bits et être valables pendant 730 jours.

Pour faciliter la distribution des certificats aux projets, une API devra être développée sur votre serveur. Cette API recevra en paramètre GET les éléments suivants :

Champ GET	**Valeur attendue**
`country`	Code de pays (exemple : `FR`)
`state`	Région (exemple : `Bourgogne`)
`locality`	Ville (exemple : `Auxerre`)
`ou`	Entité de la société (exemple : `Annuaire`)
`domain`	Sous-domaine pour lequel le certificat est généré (exemple : `annuaire.papinenergies.fr`)

Le certificat doit être retourné dans le corps de la page, avec les en-tête adaptées :

```
GET /?country=FR&state=Bourgogne&locality=Auxerre&ou=Annuaire&domain
    =annuaire.papinenergies.fr

HTTP/1.0 201 Created
Content-Type: application/x-x509-user-cert
-----BEGIN CERTIFICATE-----
<contenu du certificat>
-----END CERTIFICATE-----
-----BEGIN PRIVATE KEY-----
<contenu de la clef privée générée>
-----END PRIVATE KEY-----
```

Lors de la génération du certificat, le champ `Organization Name` doit être "`Papin Energies`" et le champ `domain` doit être un sous-domaine du domaine "`papinenergies.fr`". Si le domaine renseigné lors de l'appel de l'API n'est pas conforme, l'API doit retourner l'erreur `403 Forbidden`.

Création d'un serveur proxy filtrant Squid

Sources de la documentation :

— `https://elatov.github.io/2019/01/using-squid-to-proxy-ssl-sites/`

— `https://grumpytechie.net/2020/02/25/adding-custom-root-ca-certificates-to-debian/`

Procédure validée sur Debian 13.

Documentation disponible en ligne : https://antoinepernot.fr/articles/proxy-filtrant

Présentation du projet

Nous allons réaliser un proxy interceptant les communications entre un réseau local et Internet. Dans cette documentation, il filtrera certains domaines et certaines URL, notamment les sites pour adultes et les régies publicitaires.

La machine utilisée possédera deux cartes réseau :

enp0s3 Connectée sur Internet et dont l'adresse IP est obtenue *via* DHCP

enp0s8 Connectée sur le réseau local et dont l'adresse IP est 10.50.0.1

Notre machine agira comme routeur pour les appareils situés sur le réseau local.

Installation des paquets

Nous allons installer Squid avec support HTTPS et nftables :

```
apt -y install squid-openssl nftables
```

Configuration du système

Nous allons configurer l'interface réseau enp0s8. Pour cela, nous modifions le fichier **/etc/network/interfaces** :

```
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
        address 10.50.0.1/16
```

Appliquer ce changement :

```
systemctl restart networking
```

Il faut permettre au serveur de transmettre les paquets reçus et d'agir comme un routeur. Pour cela, ajouter la ligne suivante dans **/etc/sysctl.d/forward.conf** :

```
net.ipv4.ip_forward=1
```

Appliquer ce changement :

```
sysctl --system
```

Nous allons créer les règles de pare-feu permettant de forcer le trafic entrant depuis l'interface enp0s8 vers le proxy en modifiant le fichier **/etc/nftables.conf**. Adapter selon les autres services déjà installés :

Pensez à créer une règle pour SSH si vous utilisez ce service pour configurer votre serveur. Ne vous enfermez pas dehors !

```
#!/usr/sbin/nft -f

flush ruleset

table inet tableinet {
    chain input {
        type filter hook input priority filter; policy drop;
        iifname lo accept
        iifname enp0s8 tcp dport 3129 accept
        iifname enp0s8 tcp dport 3130 accept
        ct state {established,related} accept
    }
    chain forward {
        type filter hook forward priority filter; policy drop;
        iifname enp0s8 udp dport 53 accept
        iifname enp0s8 udp dport 67 accept
    }
    chain output {
        type filter hook output priority filter;
    }
    chain prerouting {
        type nat hook prerouting priority dstnat;
        iifname enp0s8 tcp dport 80 redirect to 3129
        iifname enp0s8 tcp dport 443 redirect to 3130
    }
    chain postrouting {
        type nat hook postrouting priority srcnat;
    }
}
```

Activer et redémarrer nftables :

```
systemctl enable nftables.service
systemctl restart nftables.service
```

Création du script de renouvellement des listes noires

Créer le répertoire qui contiendra les listes noires :

```
mkdir /etc/squid/blacklist
chown -R proxy:proxy /etc/squid/blacklist
```

La liste noire des sites pour adultes provient de l'Université de Toulouse 1 Capitole et la liste des régies publicitaires est hébergée sur `https://pgl.yoyo.org/adservers/`.

Créer le script en charge de la mise à jour des listes noires **/opt/refresh-blacklist** :

```
#!/bin/bash
wget -O /etc/blacklist/ads https://pgl.yoyo.org/adservers/serverlist
    .php?hostformat=nohtml&showintro=0&mimetype=plaintext
wget -O /tmp/adult.tar.gz ftp://ftp.ut-capitole.fr/pub/reseau/cache/
    squidguard_contrib/adult.tar.gz
cd /tmp
tar zxvf /tmp/adult.tar.gz
cp -R /tmp/adult /etc/blacklist/
rm /tmp/adult.tar.gz
systemctl reload squid
```

Configurer cron pour mettre à jour les listes noires toutes les nuits à 2 heures du matin. Pour cela, exécuter la commande suivante :

```
crontab -e
```

Ajouter la configuration suivante :

```
0 2 * * * bash /opt/refresh-blacklist
```

Exécuter le script de renouvellement des listes noires :

```
bash /opt/refresh-blacklist
```

Configuration de Squid

Squid intervenant comme intermédiaire entre le client et le serveur, l'interception des communications HTTPS impose à Squid de générer les certificats SSL pour les domaines contactés.

Nous allons créer la base de données des certificats SSL de Squid :

```
/usr/lib/squid/security_file_certgen -c -s /var/spool/squid/ssl_db -
    M 4MB
chown -R proxy:proxy /var/spool/squid
```

Sauvegarder la configuration par défaut de Squid et effacer la configuration :

```
cp /etc/squid/squid.conf /etc/squid/squid.conf.bak
echo '' > /etc/squid/squid.conf
```

Créer la configuration de Squid pour analyser le trafic de manière transparente en bloquant les URL et les domaines renseignés dans des fichiers de listes noires. Pour cela, on édite le fichier de configuration **/etc/squid/squid.conf** :

```
http_port 3128
http_port 3129 transparent
https_port 3130 intercept ssl-bump cert=/etc/squid/cert/
    squid_proxyCA.pem generate-host-certificates=on
sslcrtd_program /usr/lib/squid/security_file_certgen -s /var/spool/
    squid/ssl_db -M 4MB
ssl_bump bump all
ssl_bump peek all
acl purge method PURGE
http_access allow purge localhost
http_access deny purge
coredump_dir /var/spool/squid
refresh_pattern ^ftp:        1440    20% 10080
refresh_pattern ^gopher:     1440 0% 1440
refresh_pattern -i (/cgi-bin/|\?) 0    0% 0
refresh_pattern (Release|Packages(.gz)*)$      0       20%     2880
refresh_pattern .        0    20% 4320
acl Safe_ports port 80        # http
acl Safe_ports port 443        # https
acl CONNECT method CONNECT
acl adult dstdomain "/etc/blacklist/adult/domains"
acl adult url_regex "/etc/blacklist/adult/urls"
acl ads dstdomain "/etc/blacklist/ads"
http_access deny !Safe_ports
http_access allow CONNECT
http_access deny manager
http_access allow localhost
http_access deny adult
http_access deny ads
http_access allow all
```

Créer le certificat de l'autorité de certification de Squid :

```
mkdir -p /etc/squid/cert/
openssl req -new -days 3650 -newkey rsa:4096 -sha256 -nodes -x509 -
    keyout /etc/squid/cert/squid_proxyCA.pem -out /etc/squid/cert/
    squid_proxyCA.pem
chown -R proxy:proxy /etc/squid/cert/
chmod 0400 /etc/squid/cert/squid_proxyCA.pem
```

Ajouter le certificat de l'autorité de certification de Squid dans le magasin des autorités de certifications reconnues :

```
mkdir -p /usr/local/share/ca-certificates
openssl x509 -inform PEM -in /etc/squid/cert/squid_proxyCA.pem -out
    /usr/local/share/ca-certificates/squid_proxyCA.crt
update-ca-certificates
```

Redémarrer le service Squid :

```
systemctl reload squid
```

Installation du certificat de l'autorité de certification sur les clients

Sur le client, copier le fichier **/usr/local/share/ca-certificates/squid_proxyCA.crt** situé sur le serveur dans le répertoire **/usr/local/share/ca-certificates/** du client.

Mettre à jour le magasin des certificats :

```
update-ca-certificates
```

Serveur OpenVPN avec intégration LDAP

Sources de la documentation :

— `https://wiki.debian.org/OpenVPN`

— `https://tinyurl.com/jeu2x3fs`

— `https://kifarunix.com/configure-openvpn-ldap-based-authentication/`

Procédure validée sur Debian 13.

Documentation disponible en ligne : https://antoinepernot.fr/articles/openvpn

Présentation du projet

Nous allons mettre en œuvre, ici, un serveur OpenVPN utilisant des clefs TLS et authentifiant les utilisateurs avec un annuaire OpenLDAP.

Pré-requis

Pour la création d'un annuaire OpenLDAP, reportez-vous à la documentation spécifique : `https://antoinepernot.fr/articles/installation-openldap-nfs`

Pour la suite de cette documentation, les utilisateurs sont stockés dans `ou=Utilisateurs,dc=mondomaine,dc=local`. Les serveurs VPN et LDAP ont respectivement pour adresse IP 10.50.0.1 et IP 10.50.0.2. Notre zone réseau VPN sera 10.99.0.0/24. Le réseau a accéder via le VPN est le 192.168.1.0/24.

Installation et initialisation d'OpenVPN

Installer les paquets OpenVPN, ainsi que le serveur DNS utilisé par la zone réseau VPN :

```
apt install openvpn openvpn-auth-ldap bind9 nftables
```

Nous allons configurer l'interface réseau enp0s8. Pour cela, nous modifions le fichier **/etc/network/interfaces** :

```
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet static
        address 10.50.0.1/16
```

Appliquer ces changements :

```
systemctl restart networking
```

Autoriser le trafic vers le serveur VPN et en loopback en modifiant le fichier **/etc/nftables.conf**. Adapter selon les autres services déjà installés :

Pensez à créer une règle pour SSH si vous utilisez ce service pour configurer votre serveur. Ne vous enfermez pas dehors !

```
#!/usr/sbin/nft -f

flush ruleset

table inet tableinet {
   chain input {
      type filter hook input priority filter; policy drop;
      iifname lo accept
      iifname tun0 accept
      ip protocol icmp accept
      udp dport 1194 accept
      ct state {established,related} accept
   }
   chain forward {
      type filter hook forward priority filter;
   }
   chain output {
      type filter hook output priority filter;
   }
   chain prerouting {
      type nat hook prerouting priority dstnat;
   }
   chain postrouting {
      type nat hook postrouting priority srcnat;
      masquerade
   }
}
```

Activer et redémarrer nftables :

```
systemctl enable nftables.service
systemctl restart nftables.service
```

Il faut permettre au serveur de transmettre les paquets reçus et d'agir comme un routeur. Pour cela, ajouter la ligne suivante dans **/etc/sysctl.d/forward.conf** :

```
net.ipv4.ip_forward=1
```

Appliquer ce changement :

```
sysctl --system
```

On initialise la PKI du VPN. Vous pouvez retirer l'option `nopass` pour définir un mot de passe pour le certificat de la CA :

```
make-cadir /etc/openvpn/easy-rsa/
./easyrsa init-pki
./easyrsa build-ca nopass
./easyrsa build-server-full server nopass
```

Générer les paramètres DH :

```
./easyrsa gen-dh
```

Générer la clef d'authentification TLS :

```
openvpn --genkey secret /etc/openvpn/server/ta.key
```

Paramétrage de l'authentification d'OpenVPN

Créer le répertoire qui accueillera les paramètres d'authentification :

```
mkdir /etc/openvpn/auth
```

Créer le fichier de paramètres LDAP **/etc/openvpn/auth/ldap.conf** :

```
<LDAP>
    URL         ldap://10.50.0.2
    BindDN      cn=lecture,dc=mondomaine,dc=local
    Password    test
    Timeout     15
    TLSEnable   no
    FollowReferrals no
</LDAP>

<Authorization>
    BaseDN      "ou=Utilisateurs,dc=mondomaine,dc=local"
    SearchFilter    "(&(uid=%u))"
    RequireGroup    false
</Authorization>
```

Configuration d'OpenVPN

Définir le fichier de configuration à charger avec systemd en éditant cette ligne dans **/etc/default/openvpn** :

```
AUTOSTART="server"
```

Créer la configuration d'OpenVPN dans **/etc/openvpn/server.conf**. Modifier la/les ligne(s) `push "route x.x.x.x x.x.x.x"` en fonction des zones réseau que vous souhaitez rendre accessibles depuis le VPN :

```
local 10.50.0.1
port 1194
proto udp
dev tun
ca /etc/openvpn/easy-rsa/pki/ca.crt
cert /etc/openvpn/easy-rsa/pki/issued/server.crt
key /etc/openvpn/easy-rsa/pki/private/server.key
dh /etc/openvpn/easy-rsa/pki/dh.pem
server 10.99.0.0 255.255.255.0
ifconfig-pool-persist /var/log/openvpn/ipp.txt
push "route 192.168.1.0 255.255.255.0"
push "dhcp-option DNS 10.50.0.1"
keepalive 10 120
tls-auth /etc/openvpn/server/ta.key
cipher AES-256-CBC
persist-key
persist-tun
status /var/log/openvpn/openvpn-status.log
verb 3
explicit-exit-notify 1
push "redirect-gateway def1"
client-to-client
plugin /usr/lib/openvpn/openvpn-auth-ldap.so /etc/openvpn/auth/ldap.
   conf
```

Créer la configuration pour les clients dans **/etc/openvpn/client.conf** :

```
client
dev tun
proto udp
remote 10.50.0.1 1194
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
cipher AES-256-CBC
auth-user-pass
verb 3
```

Créer le fichier de journal d'OpenVPN :

```
touch /var/log/openvpn/openvpn-status.log
```

Redémarrer le service :

```
systemctl restart openvpn
```

Création des clefs et de la configuration client

Sur le serveur, créer les clefs pour le client (ici l'utilisateur `pdubois`). Ajouter `nopass` pour ne pas mettre de mot de passe à la clef TLS. Le mot de passe peut être passé en argument en ajoutant l'option `-passout=pass:MonMotDePasse` :

```
cd /etc/openvpn/easy-rsa/
./easyrsa --batch build-client-full pdubois
```

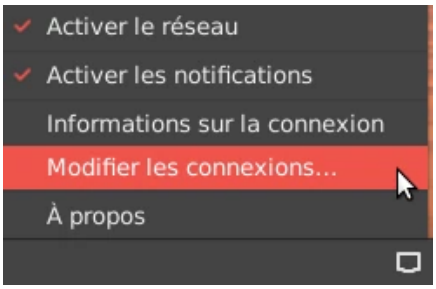
Créer le fichier de configuration **pdubois.ovpn** qui sera communiqué au client :

```
cp /etc/openvpn/client.conf /tmp/pdubois.ovpn
echo "<ca>" >> /tmp/pdubois.ovpn
cat /etc/openvpn/easy-rsa/pki/ca.crt >> /tmp/pdubois.ovpn
echo "</ca>" >> /tmp/pdubois.ovpn
echo "<cert>" >> pdubois.ovpn
sed -n '/BEGIN CERTIFICATE/,/END CERTIFICATE/p' < /etc/openvpn/easy-
    rsa/pki/issued/pdubois.crt >> /tmp/pdubois.ovpn
echo "</cert>" >> /tmp/pdubois.ovpn
echo "<key>" >> /tmp/pdubois.ovpn
cat /etc/openvpn/easy-rsa/pki/private/pdubois.key >> /tmp/pdubois.
    ovpn
echo "</key>" >> /tmp/pdubois.ovpn
echo "<tls-auth>" >> /tmp/pdubois.ovpn
sed -n '/BEGIN OpenVPN Static key V1/,/END OpenVPN Static key V1/p'
    < /etc/openvpn/server/ta.key >> /tmp/pdubois.ovpn
echo "</tls-auth>" >> /tmp/pdubois.ovpn
```

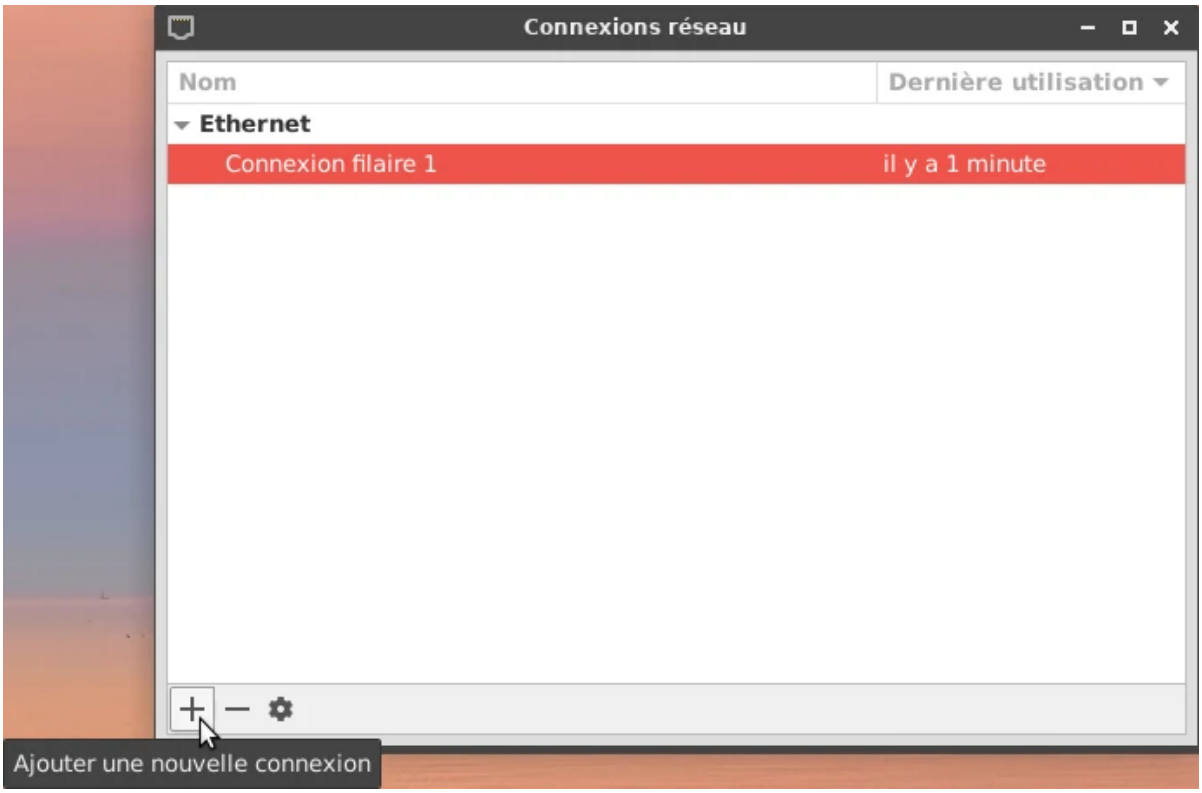
Sur le client, installer OpenVPN et le plugin OpenVPN pour Network Manager et redémarrer :

```
apt install openvpn network-manager-openvpn network-manager-openvpn-
    gnome
```

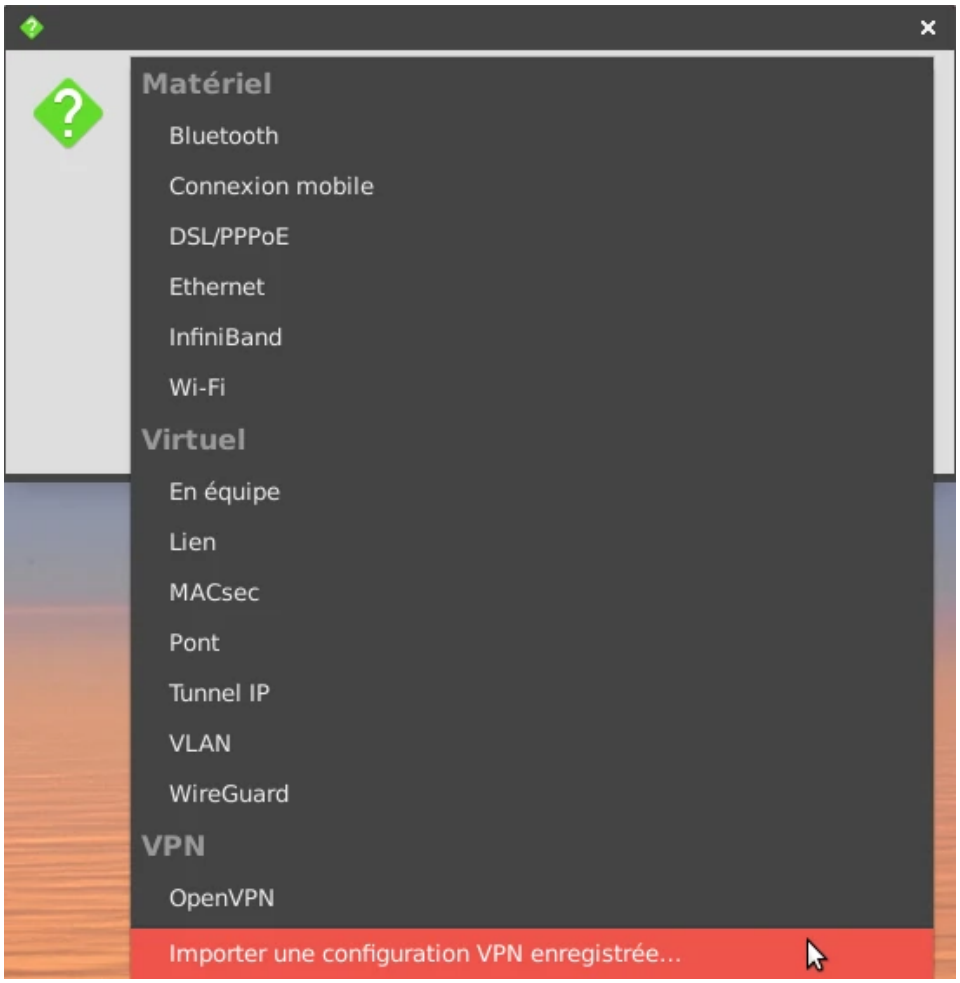
Copier le fichier **pdubois.ovpn** sur le client et le charger dans Network Manager. Pour cela, modifier les connexions :



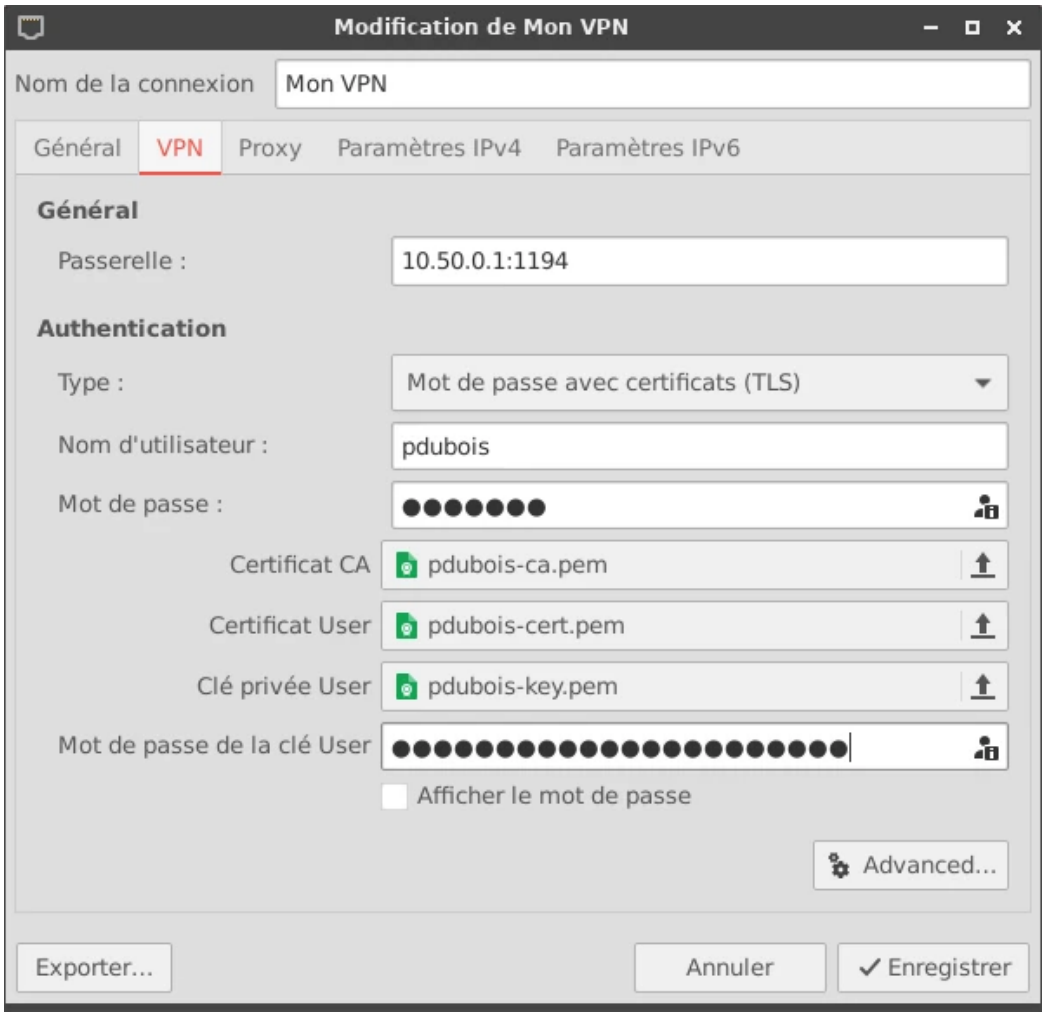
Ajouter une nouvelle connexion :



Sélectionner **Importer une configuration VPN enregistrée . . .** et sélectionner le fichier :



Modifier le nom de la connexion et saisir les identifiants :



Valider et se connecter.

Pour se connecter sans interface graphique, exécuter la commande suivante :

```
openvpn --config pdubois.ovpn
```

Révocation des clefs client

Pour révoquer une clef client, exécuter les commandes suivantes :

Créer le fichier de configuration **pdubois.ovpn** qui sera communiqué au client :

```
cd /etc/openvpn/easy-rsa/
./easyrsa --batch revoke pdubois
```

Memento Shell

Auteur : Antoine Pernot

• **Gestion des fichiers/répertoires**

Créer un répertoire (make directory) :

`-p` : Créer le répertoire parent si besoin.

```
mkdir [-p] rep
```

Créer un fichier :

```
touch fich
```

Changer de répertoire (change dir) :

```
cd rep
cd ..      # repertoire parent
cd         # repertoire personnel
cd ~alice # repertoire personnel de alice
```

Afficher répertoire courant (print working dir) :

```
pwd
```

Copier un fichier/répertoire vers un autre :

`-r` : Copier le répertoire de manière récursive.

```
cp [-r] orig dest
```

Créer un lien :

`-s` : Créer un lien symbolique.

```
ln [-s] orig lien
```

Déplacer/renommer un fichier/répertoire vers un autre (move) :

```
mv orig dest
```

Supprimer un fichier (remove) :

`-r` : Supprime un répertoire de manière récursive.

```
rm [-r] fich
```

Supprimer un répertoire vide (remove directory) :

```
rmdir rep
```

Lister le contenu d'un répertoire :

`-l` : Liste détaillée.
`-a` : Liste tous les fichiers (inclut les fichiers cachés).
`-s` : Tri par taille.
`-t` : Tri par date.
`-r` : Inverse l'ordre de tri.

```
ls [-latsr] rep
```

• **Gestion du contenu des fichiers/des flux**

Affichage brut de fichiers (non-interactif) :

```
cat [fich1 [fich2 ...]]
```

Affichage interactif de fichiers :

```
more [fich1 [fich2 ...]] # sens unique
less [fich1 [fich2 ...]] # 2 sens
```

Afficher le début d'un fichier :

`-n` : Affiche n lignes.

```
head [-n=10] [fich]
```

Afficher la fin d'un fichier :

`-n` : Affiche n lignes.

```
tail [-n=10] [fich]
```

Rechercher dans un fichier :

`-i` : Insensible à la casse.
`-v` : Affiche les lignes ne correspondant PAS à l'expression.

```
grep [-iv] expression [fich]
```

Trie les lignes d'un fichier :

`-r` : Inverse l'ordre.
`-R` : Trie dans un ordre aléatoire.

```
sort [-rR] [fich]
```

Compte les éléments d'un fichier :

`-l` : Compte le nombre de lignes.
`-w` : Compte le nombre de mots.
`-c` : Compte le nombre de caractères.

```
wc [-clw] [fich]
```

Découper un flux de texte :

`-d` : Délimiteur de découpe.
`-f` : Sélectionne les champs à renvoyer.

```
cut [-df] [fich]
```

• **Droits d'accès aux fichiers**

Changer les droits d'accès aux fichiers :

`-R` : Change les droits de manière récursive.

```
chmod [-R] {ugoa}{+-}{rwx} fich
```

Changer le propriétaire du fichier :

`-R` : Change le propriétaire de manière récursive.

```
chown [-R] nvuser[:nvgrp] fich
```

Changer le groupe du fichier :

`-R` : Change le groupe de manière récursive.

```
chgrp [-R] nvgrp fich
```

• **Recherche de fichiers**

Rechercher un fichier :

`-exec` : Exécute une commande en replaçant {} par le chemin de chacun des fichiers trouvés.

```
find rep_rech -name regex [-exec cmd {} ';']
```

• **Gestion des flux de texte**

Rediriger la sortie d'une commande vers l'entrée d'une autre :

```
cat villes.txt | grep Dijon
```

Écrire la sortie d'une commande dans un fichier (écrase le contenu) :

```
grep Dijon villes.txt > info_Dijon.txt
```

Ajoute la sortie d'une commande à un fichier :

```
grep Dijon villes.txt >> info_Dijon.txt
```

Ajoute la sortie d'erreurs d'une commande à un fichier :

```
grep Dijon villes.txt 2>> erreurs_villes.txt
```

• **Contrôle de tâches**

Affiche les processus en cours d'exécution :

```
ps aux
top     # Mode interactif
```

Envoie un signal (d'arrêt) au processus :

`-9` : Envoie un signal d'arrêt SIGKILL

```
kill [-9] pid
```

Envoie un signal (d'arrêt) aux processus appelés "nom" :

```
killall nom
```

• **Aide système**

Lister les pages de manuel contenant une chaîne de caractère :

```
apropos chaine
```

Afficher la page de manuel d'une commande :

```
man cmd
```

Memento scripts

Auteur : Antoine Pernot

- **Structure de base**

```
#!/bin/bash
# Version du script
echo "Bonjour"
exit 0
```

- **Les variables**

Affecter une variable :

```
message="Coucou !"
read rep # Stocke la reponse utilisateur
```

Appeler une variable :

```
echo $message
```

Appeler un fragment de chaîne de caractères :

```
echo ${message:offset:nbchars}
```

Variables spéciales :

$*	Contient tous les arguments passés à la fonction.
$#	Contient le nombre d'argument.
$?	Contient le code de retour de la dernière opération.
$0	Contient le nom du script.
$n	Contient l'argument n.
$!	Contient le PID de la dernière commande lancée.

- **Les tableaux**

Affectation (1ère méthode) :

```
tab=(valeur1 valeur2 ...)
```

Affectation (2ème méthode) :

```
tab[0]=John Smith
tab[1]=Jane Doe
```

Compter le nombre d'éléments du tableau :

```
len=${#tab[*]}
```

Afficher un élément :

```
echo ${tab[1]}
```

Afficher tous les éléments :

```
echo ${tab[@]}
```

- **Les structures de contrôle**

Syntaxe :

```
[ -f fichier ]
```

Opérateurs de test :

-e fichier	Contrôle si fichier existe.
-d fichier	Contrôle si fichier existe et est un répertoire.
-f fichier	Contrôle si fichier existe et est un fichier 'normal'.
-w fichier	Contrôle si fichier existe et est en écriture.
-x fichier	Contrôle si fichier existe et est exécutable.
f1 -nt f2	Contrôle si f1 est plus récent que f2.
f1 -ot f2	Contrôle si f1 est plus vieux que f2.

Opérateurs de comparaison numériques :

$n1 -eq $n2	Vérifie si les nombres sont égaux. Utiliser = pour les chaînes de caractères.
$n1 -ne $n2	Vérifie si les nombres sont différents. Utiliser != pour les chaînes de caractères.
$n1 -lt $n2	Vérifie si n1 est inférieur à n2.
$n1 -le $n2	Vérifie si n1 est inférieur ou égal à n2.
$n1 -gt $n2	Vérifie si n1 est supérieur à n2.
$n1 -ge $n2	Vérifie si n1 est supérieur ou égal à n2.

Les opérateurs logiques :

expr1 -a expr2 ou expr1 && expr2	Opérateur ET.
expr1 -o expr2 ou expr1 || expr2	Opérateur OU.
! expr	Opérateur NON.

- **Les conditions**

```
if [ condition1 ]
then
    Condition 1 vraie
elif [ condition2 ]
then
    Condition 2 vraie
else
    Aucune condition vraie
fi
```

- **Les boucles**

"Tant que ..." :

```
while [ condition ]
do
     Tant que la condition est vraie
done
```

"Pour ..." :

```
for variable in valeurs
do
    instructions
done
```

- **Les aiguillages**

```
case "$var" in
    val1 | val2 )  Valeur 1 ou 2;;
    val3 )         Valeur 3;;
    * )            Autres valeurs;;
esac
```

- **Les fonctions**

Déclaration de la fonction :

```
nom_fonction(){
        instructions
}
```

Appel de la fonction :

```
nom_fonction
```

- **La couleur**

Syntaxe :

```
echo -e '\033[A;B;Cm Bonjour ! \033[0m'
```

Valeurs d'effets (A) :

0	Normal
1	Gras
21	Non-gras
4	Souligné
24	Non souligné
5	Clignotant
25	Non-clignotant
7	Inversé
27	Non-inversé

Valeurs de couleur (B et C) :

Couleur	Couleur texte (B)	Couleur fond (C)
Noir	30	40
Rouge	31	41
Vert	32	42
Jaune	33	43
Bleu	34	44
Magenta	35	45
Cyan	36	46
Blanc	37	47

Memento nftables

Auteur : Antoine Pernot

• Structure du fichier /etc/nftables.conf

```
#!/usr/sbin/nft -f
flush ruleset
# Exemple table IPv4 et IPv6
table inet nomTable {
 # Chaine de flux entrant
 chain input {
  type filter hook input priority filter;
  # Refuser tout trafic entrant par defaut
  policy drop;
  # Inserer les regles entrantes ici
  # Exemple : tcp dport { 80, 443 } accept
  # Autoriser trafic pour
  # les sessions deja etablies
  ct state {established,related} accept
 }
 # Chaine de flux en transit vers une autre
 # destination
 chain forward {
  type filter hook forward priority filter;
 }
 # Chaine de flux sortant
 chain output {
  type filter hook output priority filter;
 }
 # Chaine avant routage
 chain prerouting {
  type nat hook prerouting priority dstnat;
 }
 # Chaine apres routage
 chain postrouting {
  type nat hook postrouting priority srcnat;
 }
}
```

• Tables

Type	Description
ip	IPv4 (par défaut)
ip6	IPv6
inet	IPv4 et IPv6
arp	ARP (ex arptables)
bridge	Bridge (ex ebtables)
netdev	Interface unique

• Construction règles

```
<filtre> [filtre ...] <action>
```

• Filtres courants `ip`

Filtrer par protocole :

```
ip protocol tcp
ip protocol { tcp, icmp, udp }
```

Filtrer par adresse source :

```
ip saddr 10.0.0.1
ip saddr { 10.0.0.1, 10.0.0.2 }
ip saddr 10.0.0.1-10.0.0.250
ip saddr 10.0.0.0/24
ip saddr != 10.0.0.254
```

Filtrer par adresse destination :

```
ip daddr 10.0.0.1
```

• Filtres courants `ip6`

Filtrer par adresse source :

```
ip6 saddr fe80:cafe::beef
ip6 saddr { fe80::1, fe80::2}
ip6 saddr fe80::1-fe80::250
ip6 saddr fe80:cafe::/64
ip6 saddr fe80:cafe::acdc
```

Filtrer par adresse destination :

```
ip6 daddr fe80:cafe::beef
```

• Filtres courants `tcp` **et** `udp`

Filtrer par port source :

```
tcp sport 10250
udp sport { 53, 69 }
tcp sport { 8080-8089 }
udp saddr { dhcp, dns }
```

Filtrer par port destination :

```
tcp dport 22
```

• Ping `icmp` **et** `icmpv6`

```
icmp type echo-request accept
icmpv6 type echo-request accept
```

• Filtrer les connexion établies

```
ct state { established, related }
```

• Filtrer selon les interfaces

Filtrer par interface d'entrée :

```
iifname lo
iifname { eth0, eth1 }
iifname != wlan0
```

Filtrer par interface de sortie :

```
oifname eth0
```

• Actions

Accepter :

```
accept
```

Refuser :

```
drop
```

Limiter le trafic :

```
limit rate 1024 mbytes/second
```

• Actions pour `prerouting`

Redirection de port :

```
redirect to 8080
```

Exemple :

```
iifname eth0 tcp dport 80 redirect to 8080
```

NAT destination :

```
dnat to 10.0.0.1:8080
```

Exemple :

```
tcp dport 8080 dnat to 10.0.0.1:80
```

• Actions pour `postrouting`

NAT source :

```
masquerade
```

• Gérer le service nftables

Activer nftables au démarrage :

```
systemctl enable nftables
```

Recharger les règles :

```
systemctl restart nftables
```

Lister les règles actives :

```
nft list ruleset
```