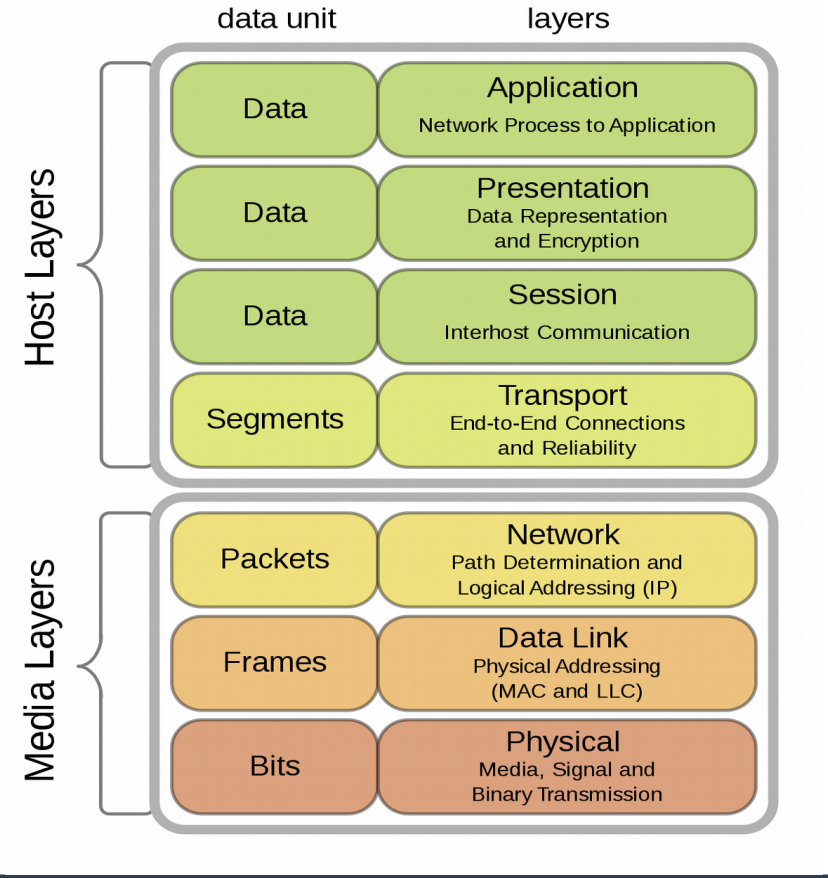
Sensibilisation à la Sécurité Informatique

Plan de l’intervention

- **Quelques repères**
- **Sécurité sur internet :** l’individu internaute, sa vie privée, l’intégrité et le secret de ses communications
- **Sécurité des infrastructures et des systèmes :** approche de concepts techniques par l’exemple (scénario fictif d’espionnage industriel)

Quelques repères

Le modèle OSI (Open Systems Interconnection)

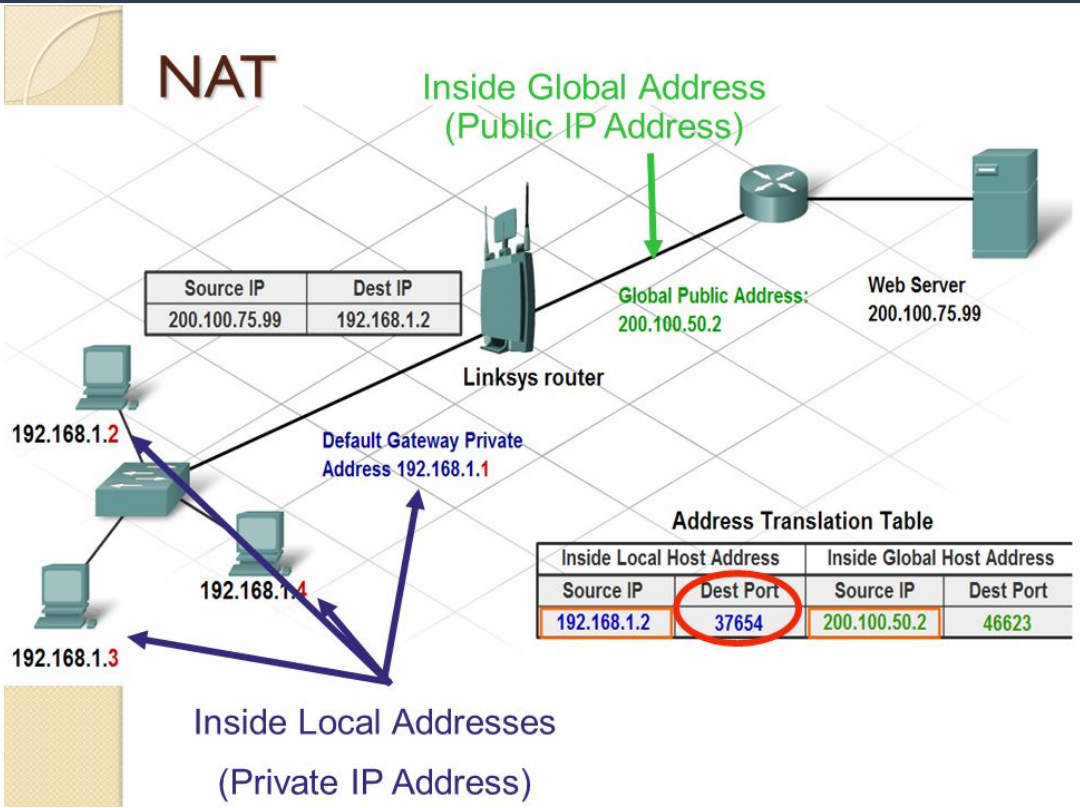


- Modèle de communication entre ordinateurs
- Les failles et exploit peuvent intervenir à différents niveaux d'abstraction :

  - failles de « haut niveau » : failles applicatives (XSS, injection SQL, CSRF, etc)

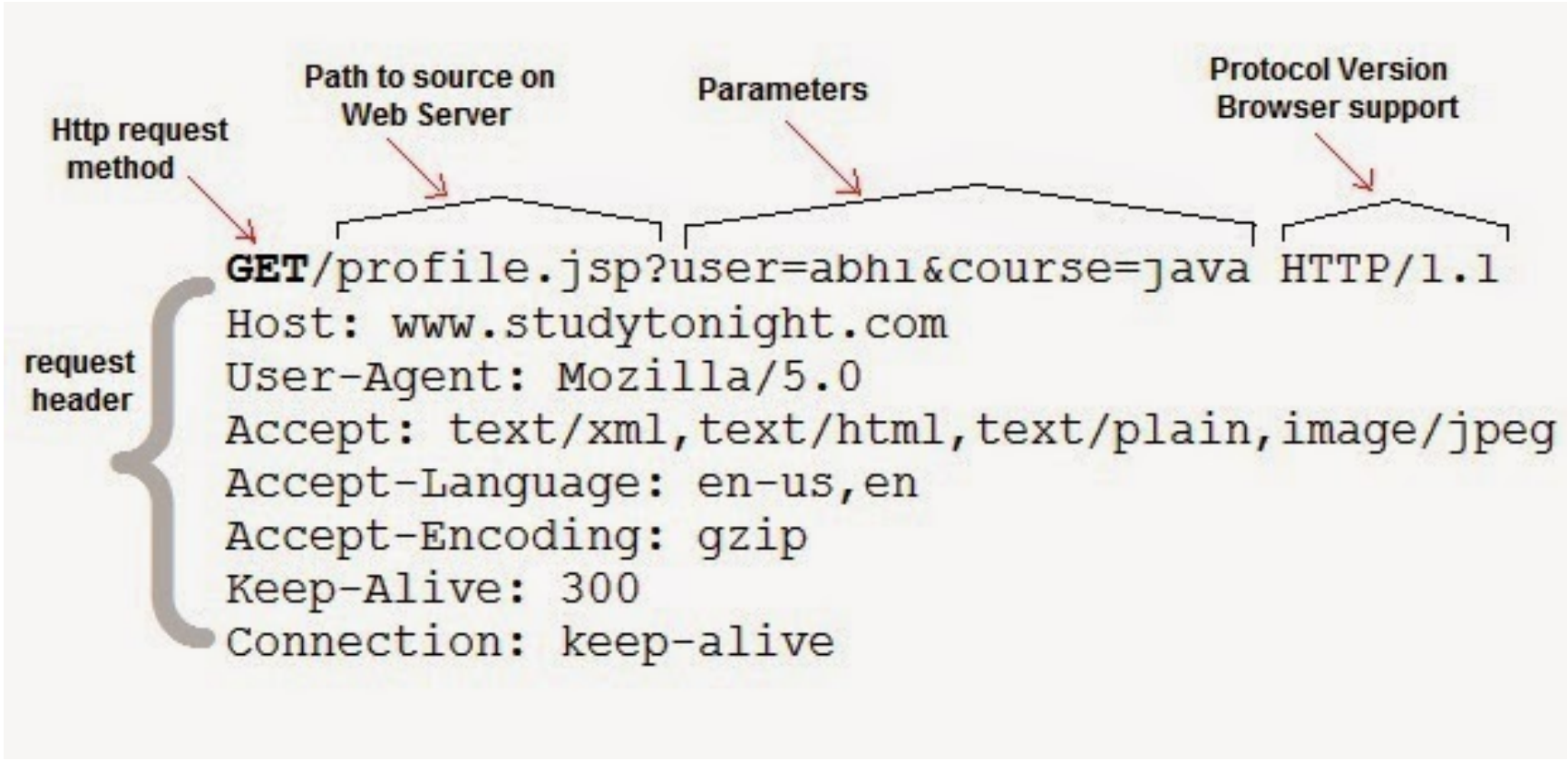
  - failles de « bas niveau » : usurpation d'identité du routeur, fuzzing, etc, etc

Le système d’adressage du protocole TCP/IP



- **Ip locales VS publiques**
- **65536 ports (codés en 16 bits)**

Qu’est-ce qu’une requête HTTP ?



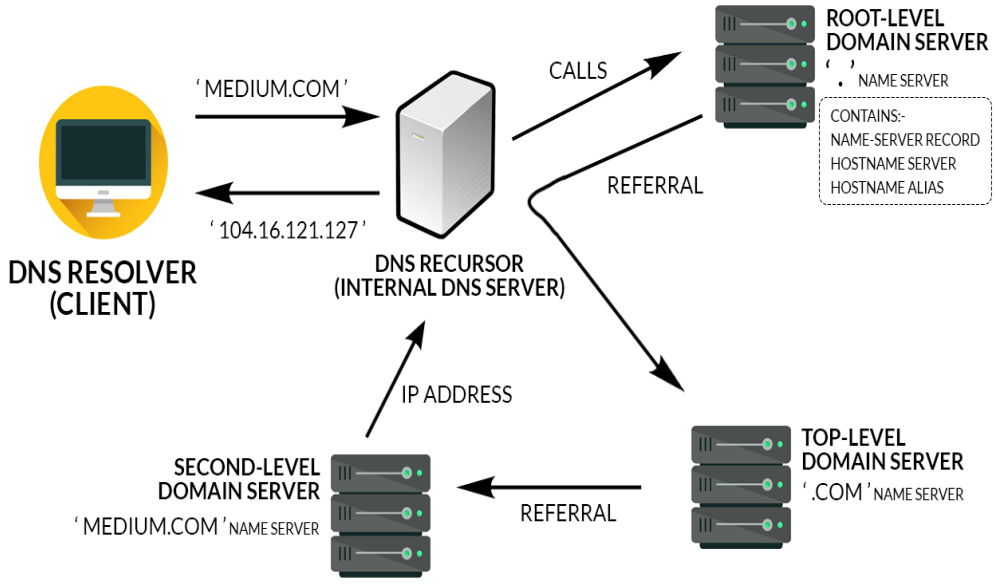
Sécurité sur internet

Que se passe-t-il lorsque je cherche « medium.com » ?

Une requête est adressée au réseau de serveurs DNS.

Fonction d'un serveur DNS :

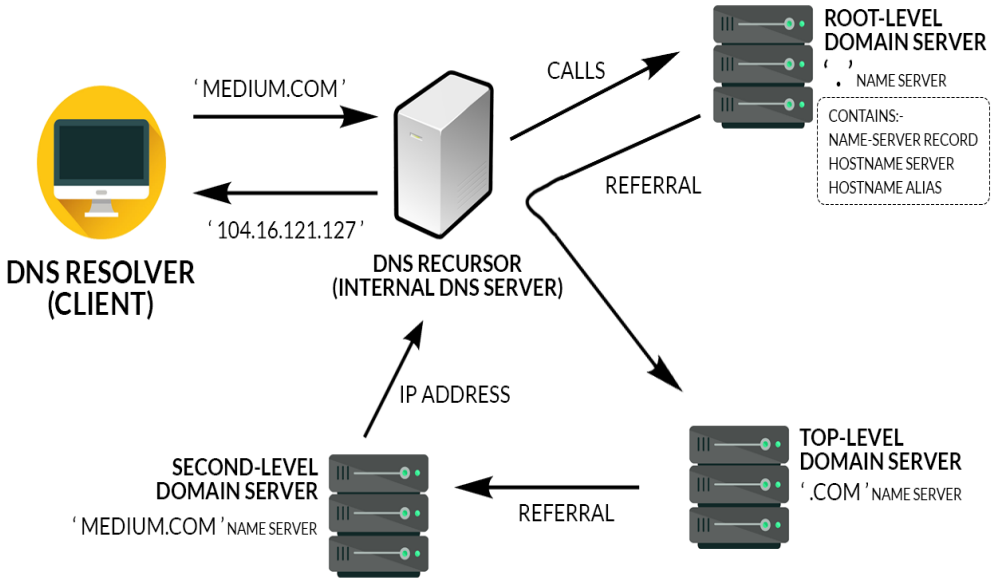
- Renvoyer l'**ip publique** associée à un nom de domaine
- S'il ne la connaît, passer la requête à un serveur DNS en amont



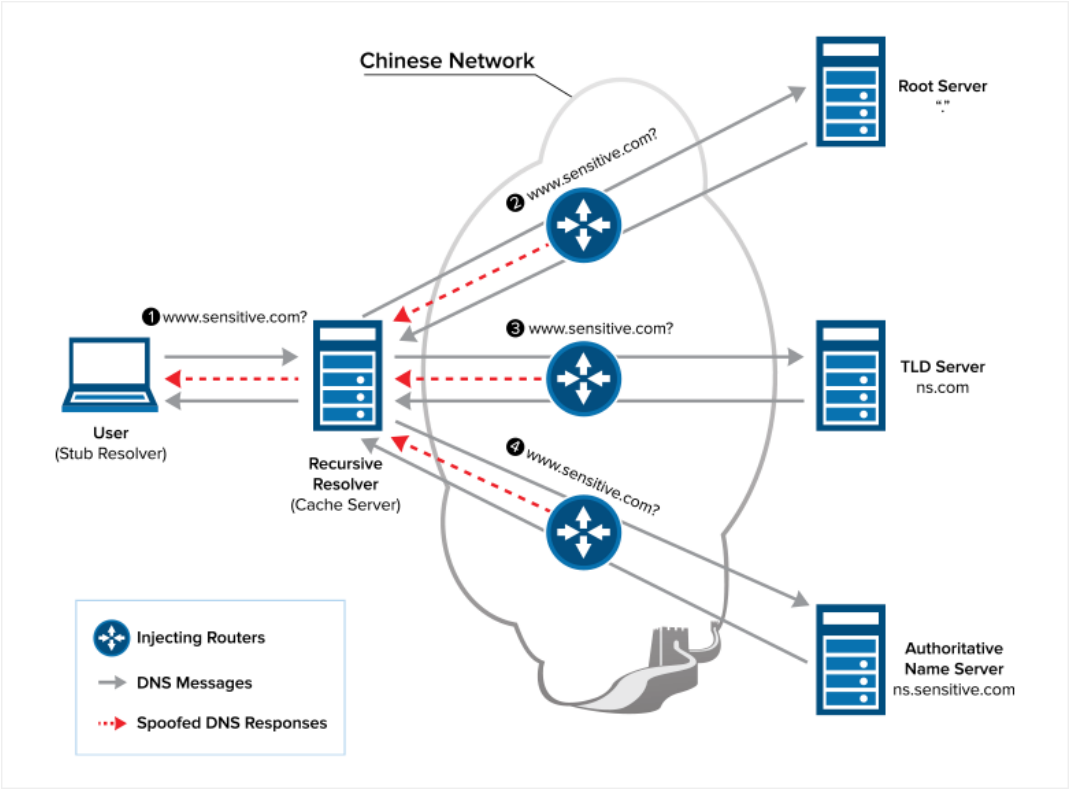
Que se passe-t-il lorsque je cherche « medium.com » ?

Pour se faire passer pour « medium.com », il suffirait d'altérer en chemin la réponse du serveur DNS pour renvoyer le client sur une autre ip publique (celle d'un site pirate)

On appelle cette technique du **DNS poisoning**



Une illustration à grande échelle : la « Grande Muraille de Chine »



Cette technique est celle utilisée par le gouvernement chinois, avec l'aide des FAI, pour empêcher l'accès à certains sites occidentaux

Comment s’assurer alors de l’identité du serveur ?

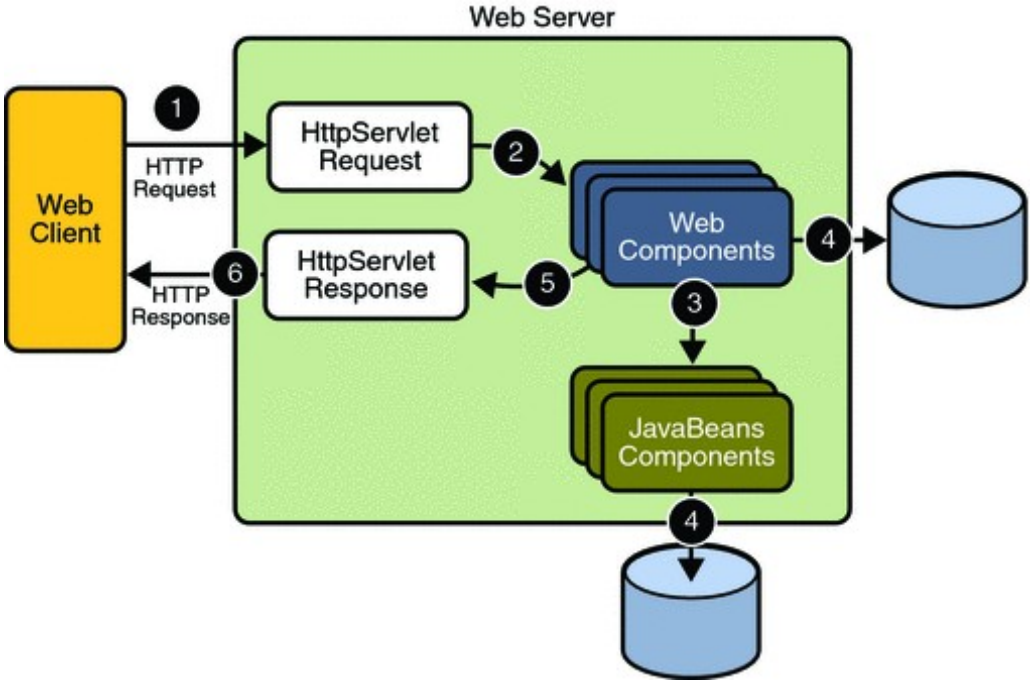
Il s’agit de la principale fonction des **certificats** (SSL notamment, mais pas exclusivement).

Nous y reviendrons lorsque nous parlerons du protocole HTTPS

Lorsque les devs n’ont pas bien fait leur boulot : l’injection SQL

Principe général

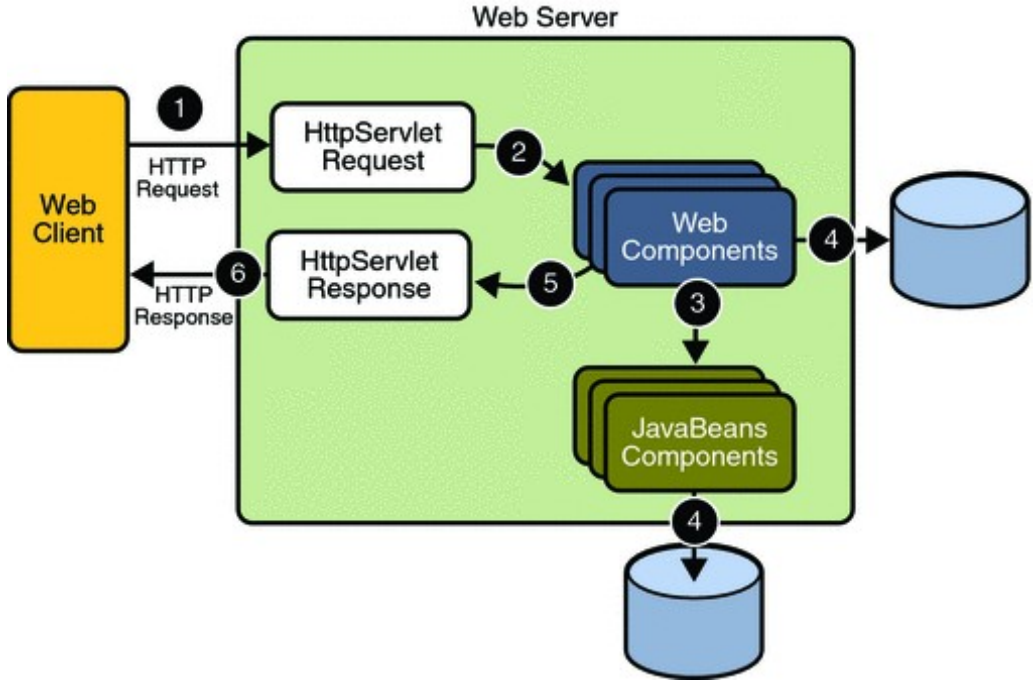
- Dissimulation de requêtes sql dans des formulaires envoyées en requêtes HTTP
- Il est possible de récupérer le contenu de la base de données (BDD) d’une application mal sécurisée



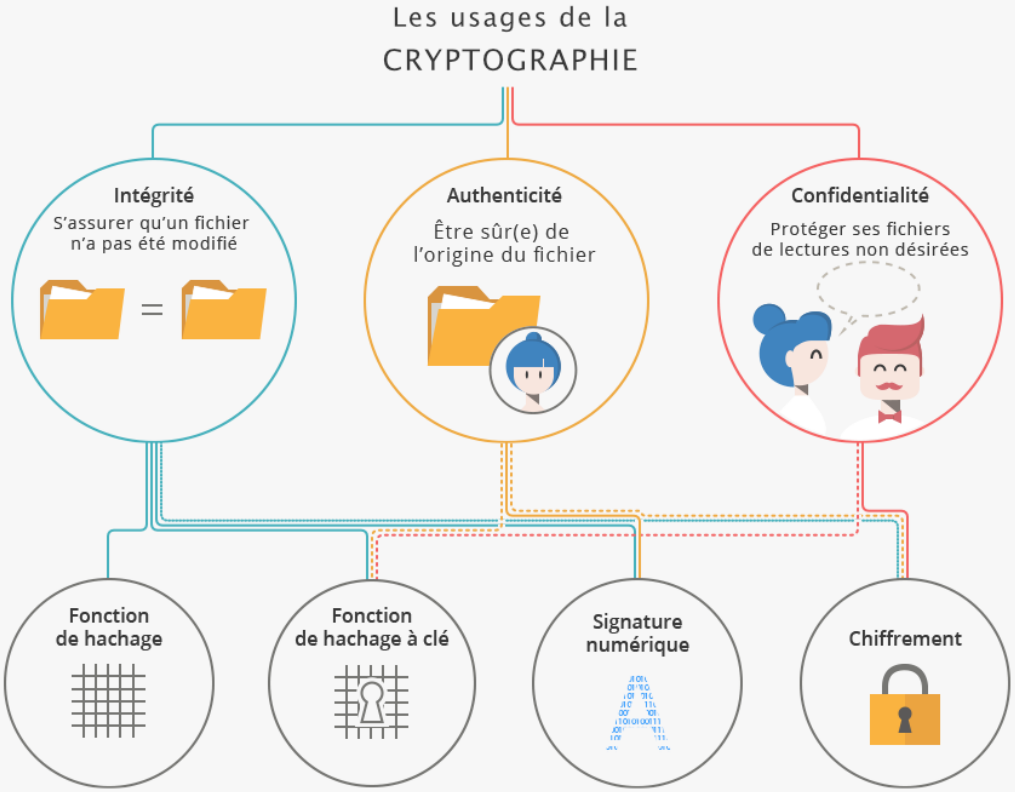
Lorsque les devs n’ont pas bien fait leur boulot : l’injection SQL

Le problème

- Dans cette BDD, il y a nécessairement une table avec les **identifiants** et **mots de passe** des utilisateur
- Sont-ils utilisables ? A priori non car les mdp sont hashés



Petit détour par la cryptographie



Il existe 2 grandes familles d'algorithmes cryptographiques (2 fonctions principales)

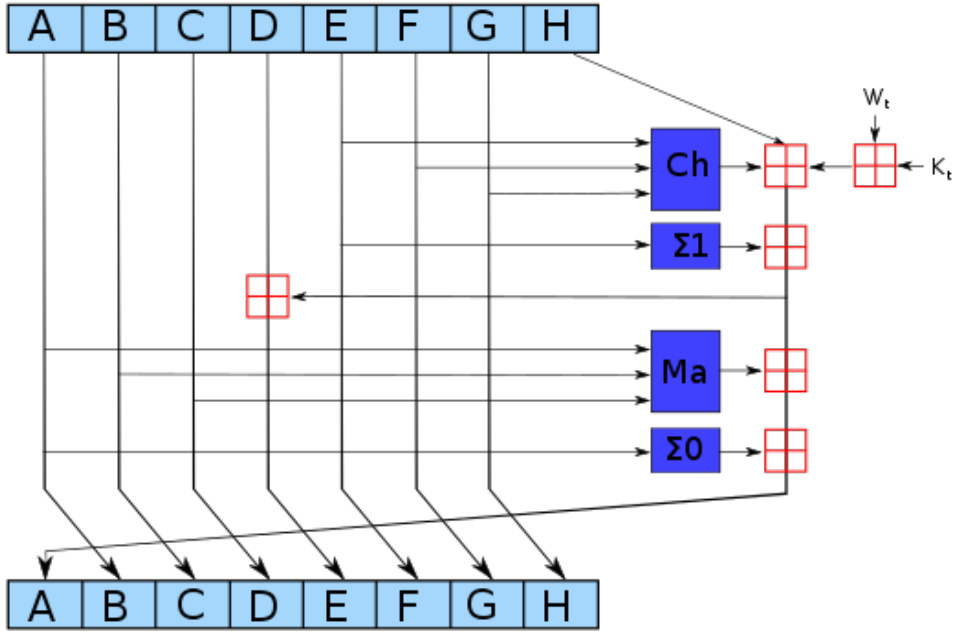
- Algos « réversibles » : chiffrement
- Algos « irréversibles » : fonction de hachage

Les fonctions de hashage (MD5, SHA256)

- Fonctions bit a bits
- Indépendantes de la longueur de l'input (fonctionnement par tour)
- Produisent une **signature unique** de longueur fixe
- Comparons :

  - echo -n foobar | sha256sum

  - echo -n Foobar | sha256sum

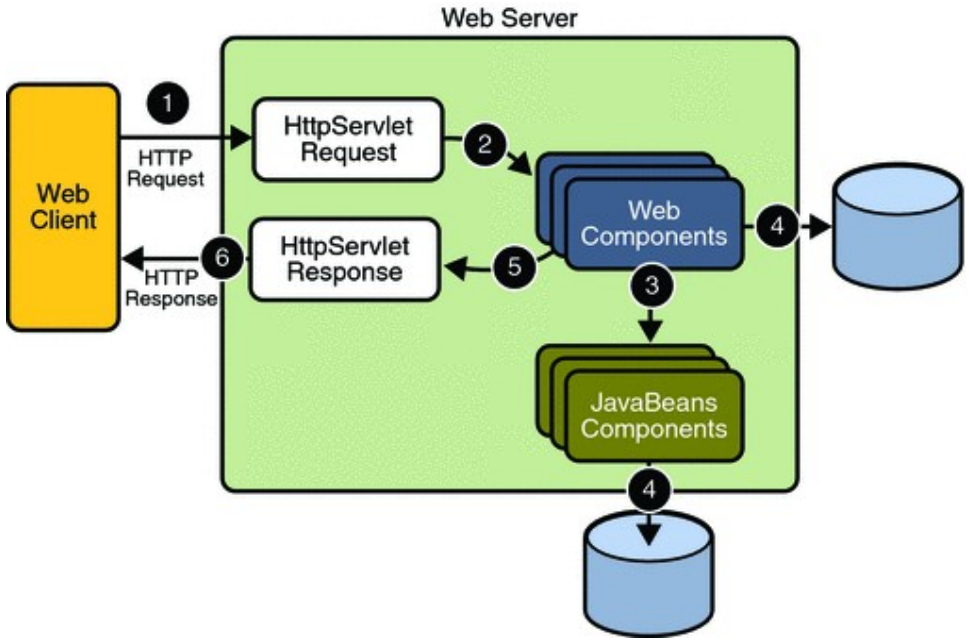


Revenons à notre scénario d'injection SQL

- A priori, les mdp récupérés par le hacker sont inutilisables
- ….a moins que le mot de passe soit :

  - trop court

  - déjà présent sur un dictionnaire



Comparons quelques ordres de grandeur

Erreur n°1 choisir un mot de passe « sémantique ».

De grandes chances de se trouver sur une liste de mdp courants :

rockyou.txt :

10,000,000 possibilités

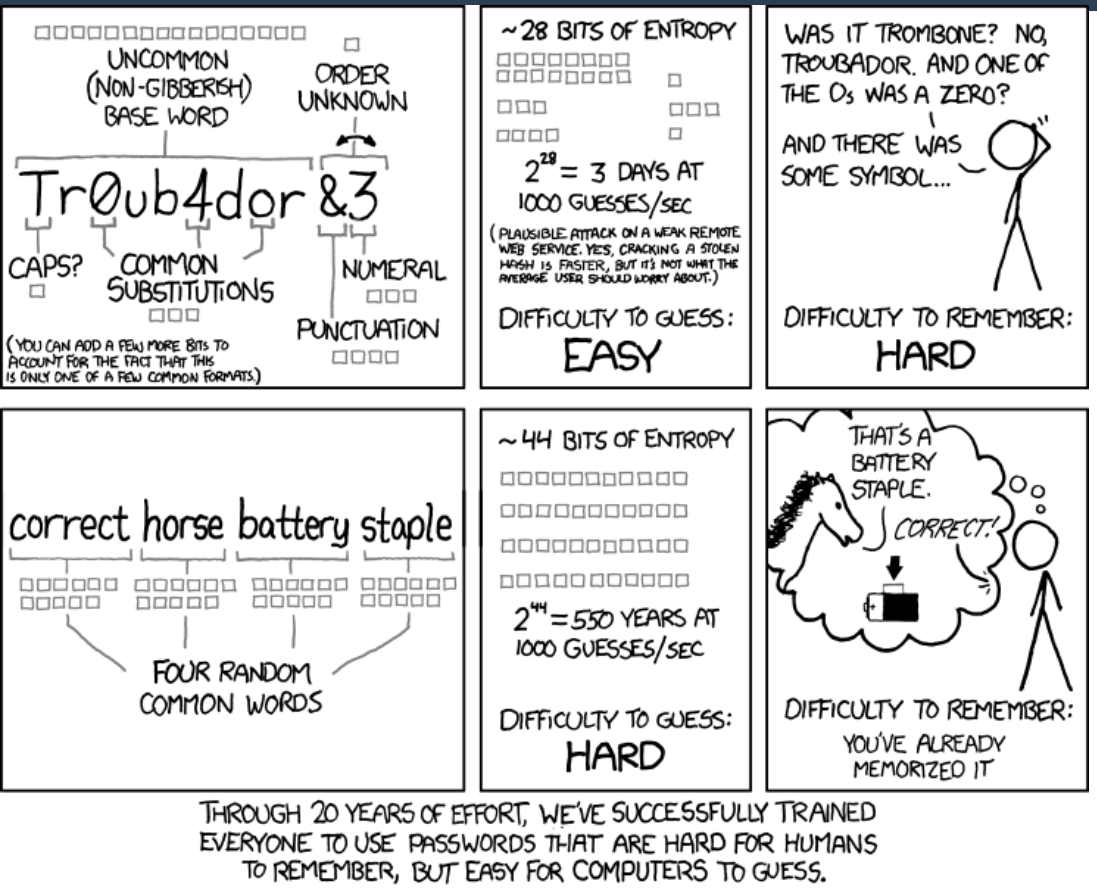
Erreur n°2 mdp trop court (ex. 4 caractères alphabétiques)

4**26 = 4,503,599,627,370,496

Respecter une longueur raisonnable (ex. 10 caractères)

100,000,000,000,000,000,000,000

Un mythe répandu sur les mots de passe



Les substitutions intuitives ne sont pas une protection

l'ajout de caractères spéciaux non plus

La vrai variable pertinente : la longueur

Ces informations envoyées ne pourraient-elles pas être interceptées « en chemin » ?

Les protocoles chiffrés :

- **HTTPS**
- **SSH**
- **FTPS**

Tout l’information transitant entre le client et le serveur est chiffré

Focus sur HTTPS : Une distinction préalable

Chiffrement symétrique

- Une seule clé crypte et décrypte
- **eg. AES 256**
- **Comment envoyer la clé au destinataire pour qu'il puisse décrypter ?**
- **Rapide**

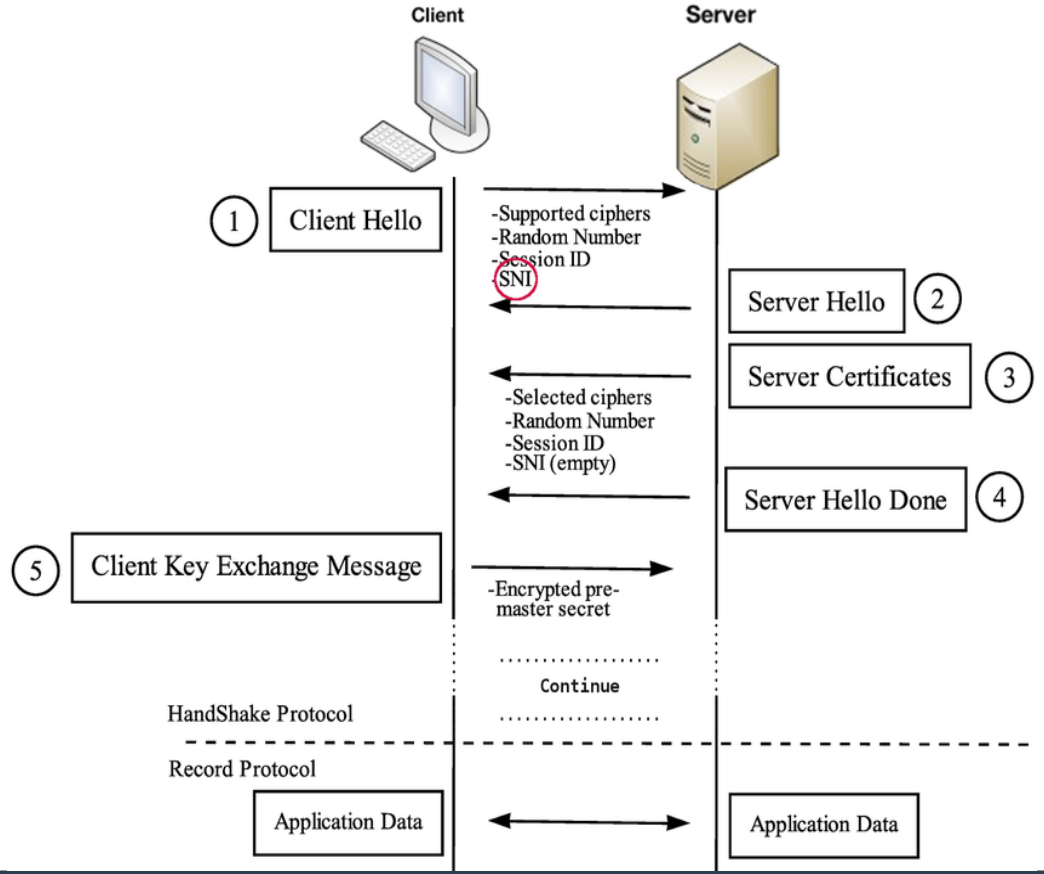
Chiffrement asymétrique

- Clé publique VS clé privé
- **eg. RSA**
- **Très solide + identité unique**
- **Très énergivore**

Focus sur HTTPS : principe du TLS Handshake

Étape 1

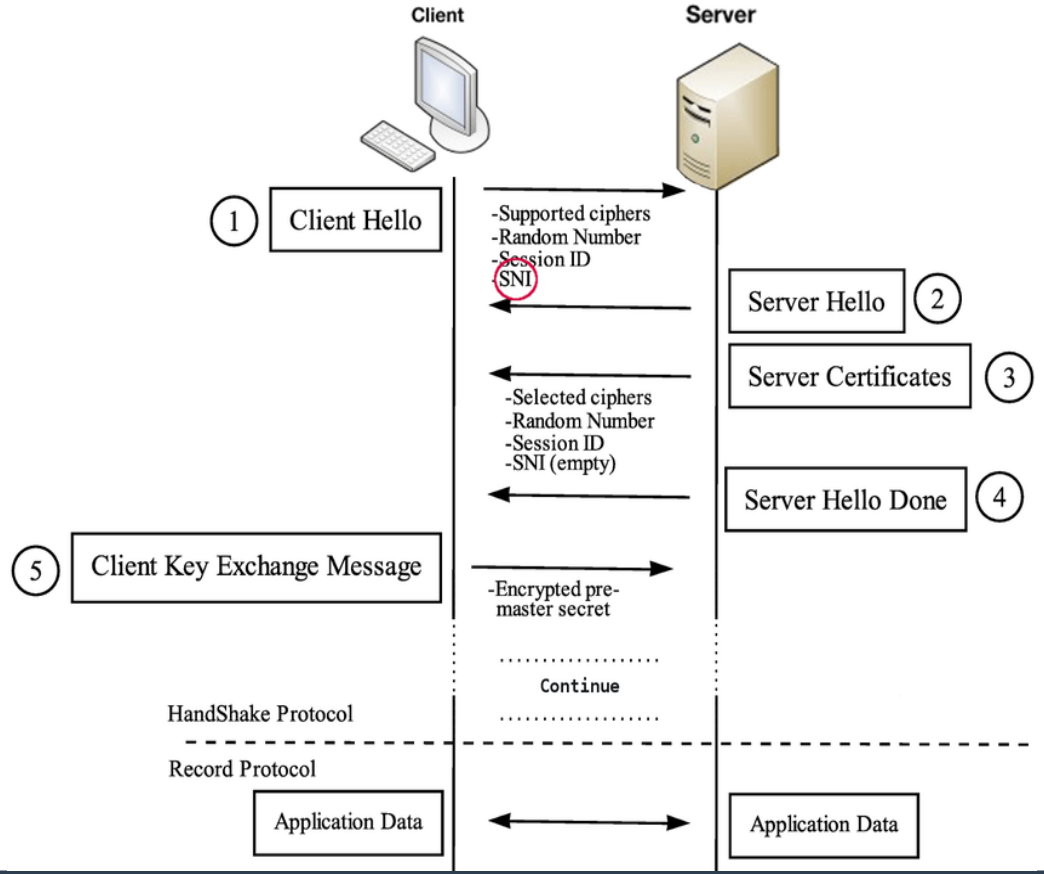
« salut Serveur, moi c’est Client. Je peux encrypter dans telle et telle langue. Voici un nombre random et l’ID de notre conversation (session). Voici le nom que je vais te donner (SNI) »



Focus sur HTTPS : principe du TLS Handshake

Étape 2-3

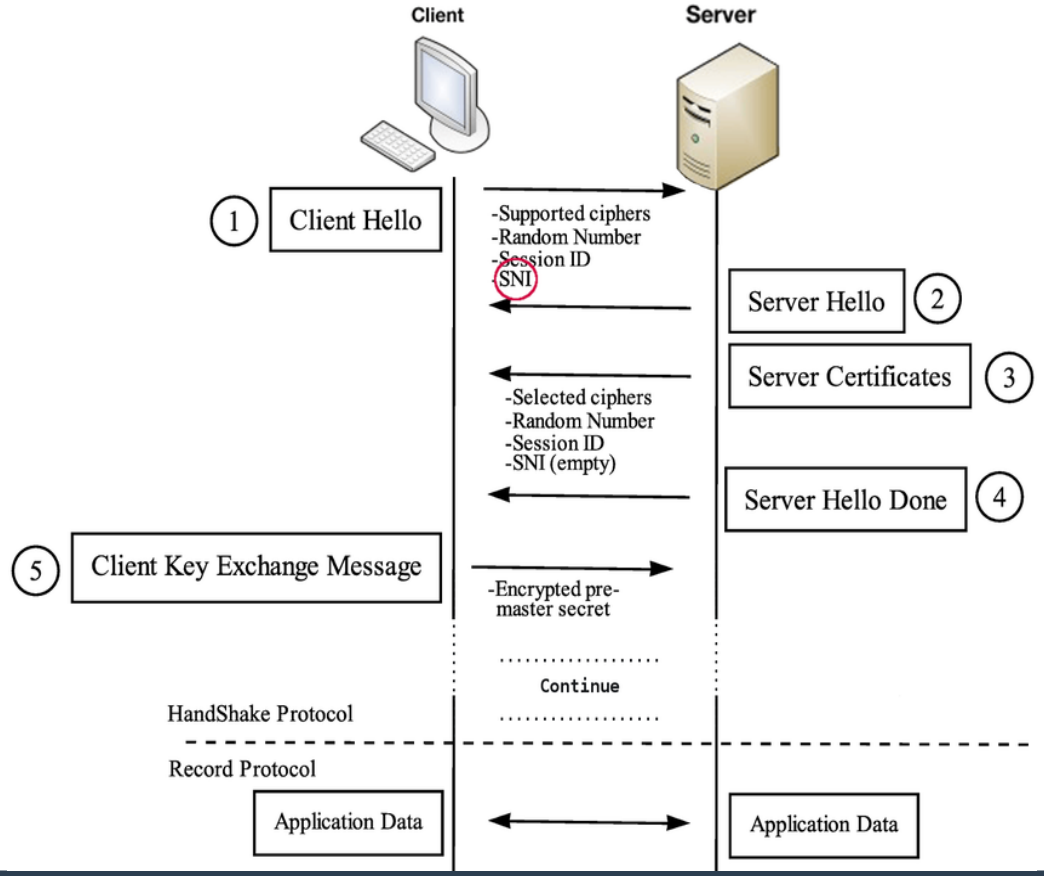
« Salut Client, bah écoute, je peux te parler dans telle langue. Tiens, vérifie que le nombre et l'ID que je te renvois correspondent à ce que tu m'as transmis »



Focus sur HTTPS : principe du TLS Handshake

Étape 4

« c'est bon, j'ai terminé »

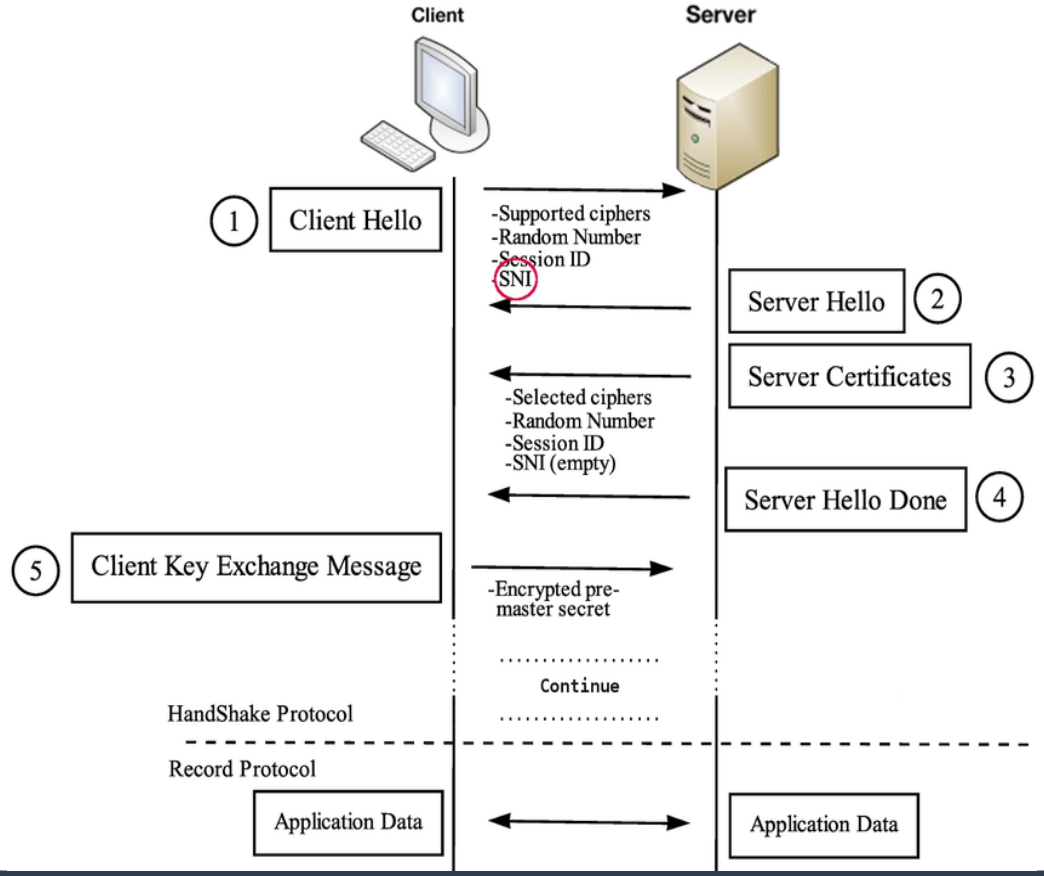


Focus sur HTTPS : principe du TLS Handshake

Étape 5

« Tiens, voilà ma clé » (encryptée avec la clé publique du serveur)

→ c'est ici que le chiffrement asymétrique intervient



Enfin, parlons d’anonymat

Il existe des contextes sensibles dans lesquels la seule confidentialité ne suffit pas :

- Lanceurs d’alerte
- Journaliste
- Politique

On aimerait dans ces contextes qu’il soit impossible de :

- Pouvoir traquer une requête jusqu’à son point d’origine
- Pouvoir associer cette requête à un nom

Comment se fait-il que je ne sois pas anonyme ?

Que voit-on de moi ?

- Niveau applicatif : cookies
- Niveau protocole :

  - user agent

  - OS

  - type de device
- Niveau réseau : mon ip publique

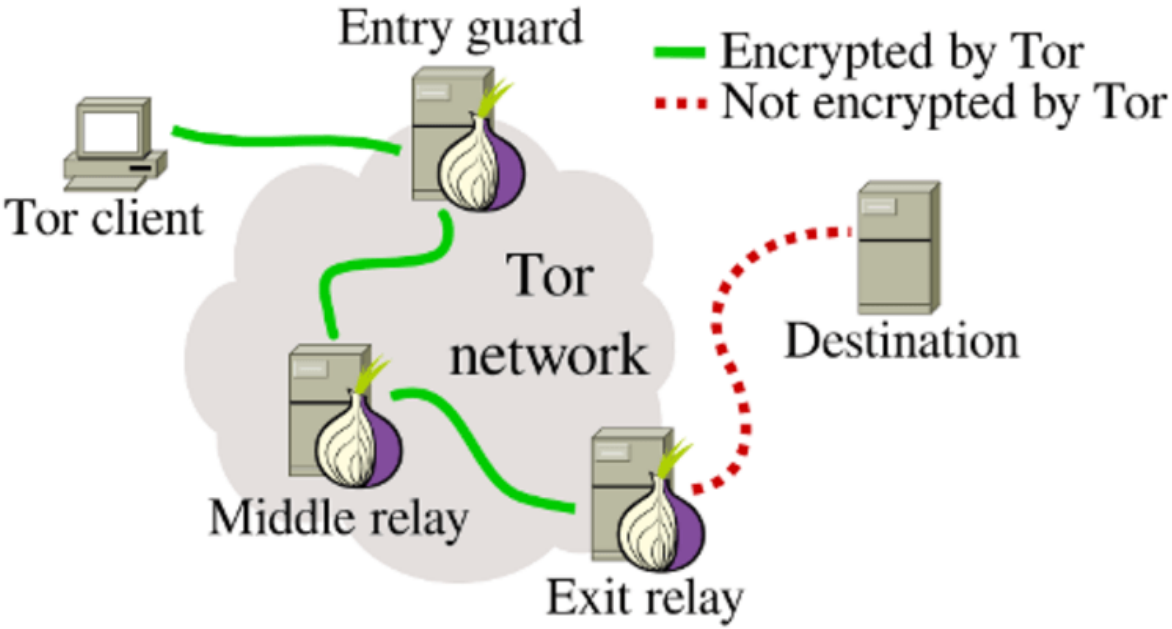
Regardons par niveau

Atelier : Changer son user agent

Atelier : installer et tester Tor

Principe général :

- Un réseau de VPN
- Redirection dynamique



Sécurité des infrastructures et des systèmes

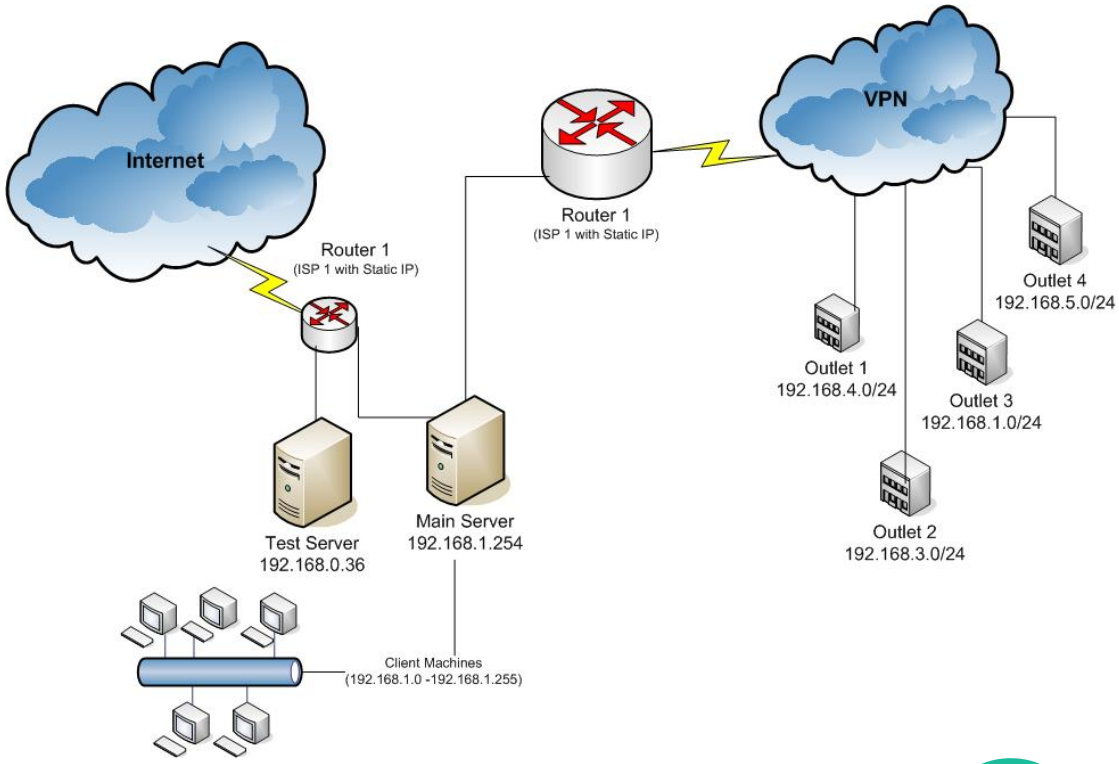
Un scénario « relativement » réaliste

Le 5 Décembre 2016, le plan d’un prototype développé par la société AnomX est diffusé sur un site pirate.

Panique, personne n’a rien vu venir. Essayons de comprendre ce qui s’est passé.

Un mot sur AnomX

- 60 employés (et le parc informatique qui va avec)
- Un serveur assurant :
  - service web
  - messagerie Web
  - serveur de MAJ
  - File sharing (FTPS)
- Un firewall balèze



Pare-feu, kesako ?

Assure la sécurité d'un réseau par le contrôle des flux de données entrantes et sortantes

- Matériel ou logiciel
- Idée générale : filtrage
- Ex : empêche les Trojans d'envoyer des informations à l'extérieur
- Sépare le réseaux interne du réseau externe
- Éventuellement, sépare plusieurs zones au sein d'un même réseau local

Selon quelle logique de filtrage ?

- Ip
- Port (TCP, UDP)
- Entrant VS sortant
- Utilisateurs
- Données elles mêmes !

- **Réseau interne → zone de confiance**
- **Zone démilitarisée DMZ**

La première piste évidente

Faille d’un service ouvert sur l’extérieur

Mais l’admin sys est formel :

- Rien de suspect dans les logs
- Parfaite intégrité des registres du serveur
- Administration du serveur pas SSH (clefs RSA)
- Rien dans les échanges avec l’extérieur ne montre de fuite provenant du serveur

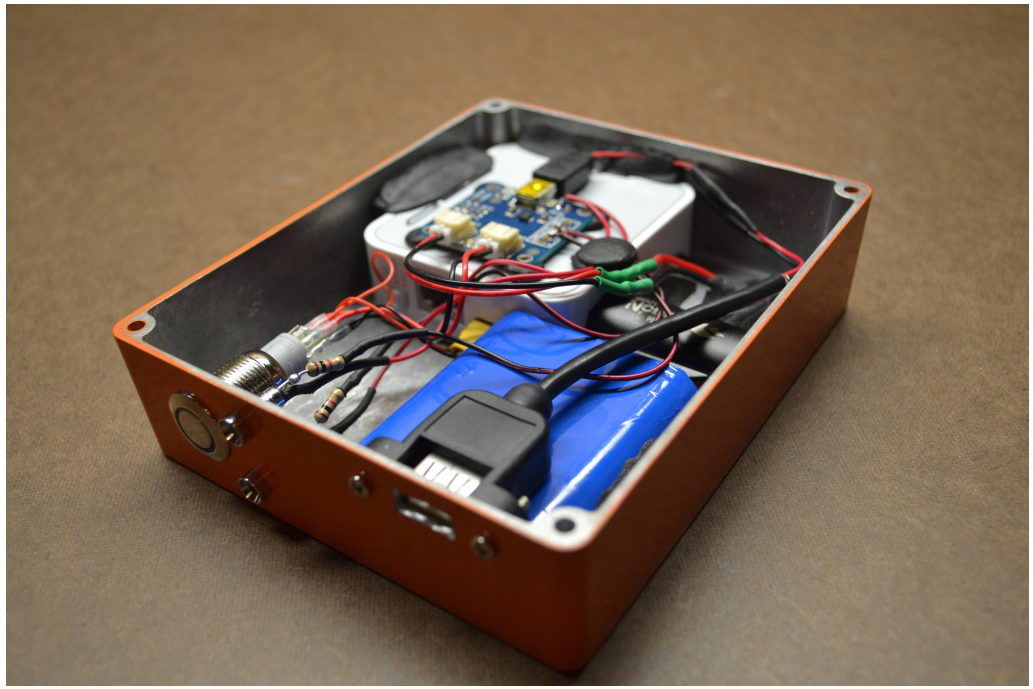
???????

Étape 1 : rentrer sur le réseau local (crack d’un protocole WPA)

3 Mai 2015

Le hacker, en visite dans les locaux, dépose ce dispositif. Sa fonction :

- envoyer des signaux de dés-authentification aux clients wifi du point d’accès le plus proche
- Récupérer les paquets d’authentification

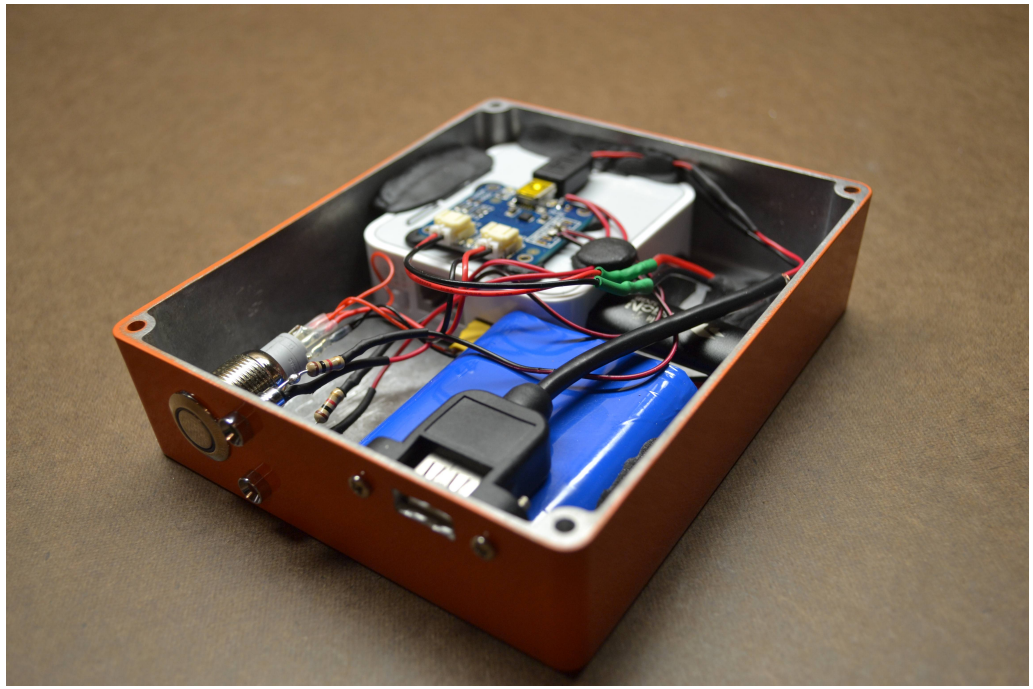


Étape 1 : rentrer sur le réseau local (crack d’un protocole WPA2)

10 Mai 2015

Le hacker, récupère le dispositif.

Certes, la clé wifi n’est pas contenue en claire dans les paquets chiffrée en CCMP (un genre d’AES) à l’aide de la PSK (Pre Shared Key)

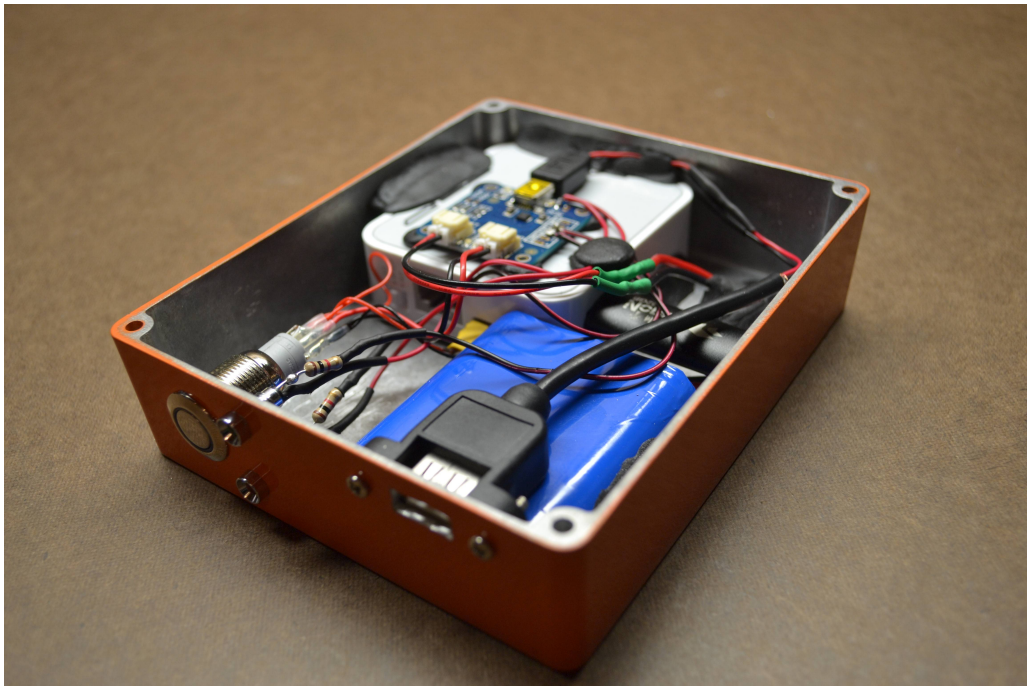


Étape 1 : rentrer sur le réseau local (crack d’un protocole WPA2)

Mai 2015 ↔ Janvier 2016

Le hacker craque la clef en louant un cluster de calcul sur Amazon AWS

(quelques milliers d’euros et de la patience)



Etape 2 : zombifier l’un des postes de travail de l’infrastructure.

Le hacker dispose désormais de la clef wifi nécessaire pour se connecter au réseau local

Il peut désormais voir les paquets échangés sur sa section du réseau local

Wireshark

Mais voler les données en essayant d’attaquer le serveur de fichier serait :

- Voué à l’échec (FTPS)
- Aussi discret qu’un sapin de Noël
- Nécessite d’être sur place

Etape 2 : zombifier l’un des postes de travail de l’infrastructure.

Mieux vaut infecter l’un des postes disposant des droits d’accès sur le service FTPS.

Après analyse des paquets de capture, le hacker a identifié :

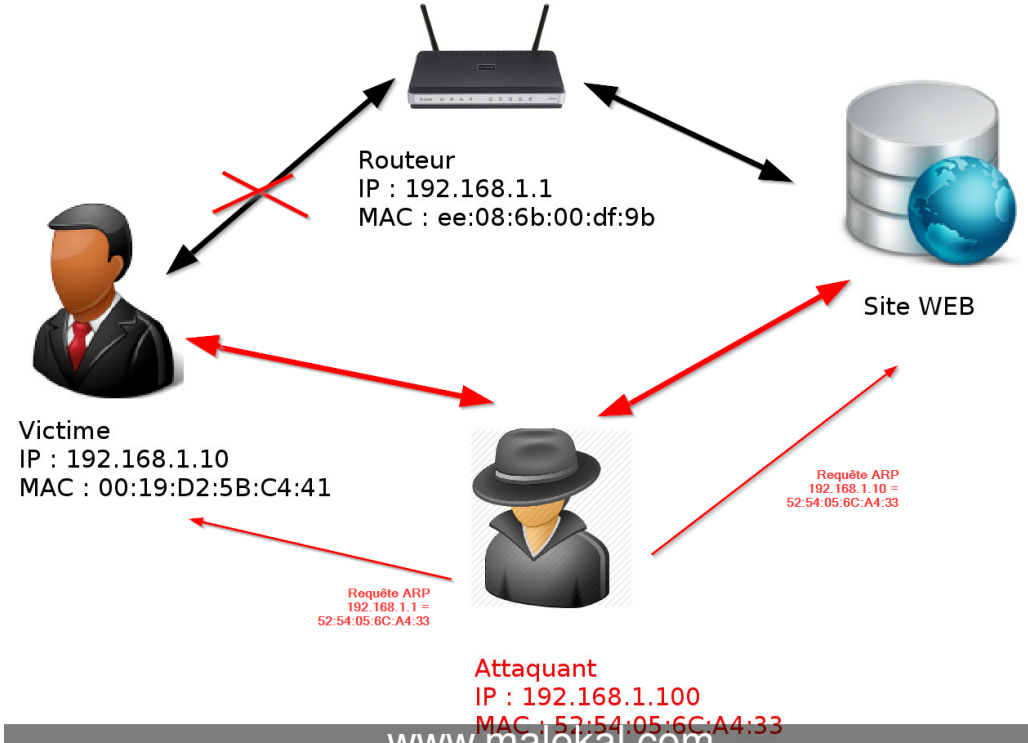
- Plusieurs postes accédant à ce service
- L’existence d’un serveur de mise a jour centralisé et potentiellement vulnérable

Etape 2 : zombifier l’un des postes de travail de l’infrastructure.

L’idée générale :

- Se placer entre la victime et le serveur de MAJ en se faisant passer pour le routeur
- Remplacer les paquets legit de MAJ par des versions modifiées

Man in the Middle



Etape 2 : zombifier l’un des postes de travail de l’infrastructure.

Le malware inséré dans le paquet de MAJ est un exécuteur de shell distant passant discrètement par le port 53

En ce beau matin de Mars 2016, le hacker reçoit un paquet indiquant que sa victime est connectée

Etape 3 : récupération et exfiltration des plans

Après récupération des info de connexion au service FTPS, le hacker télécharge les plans sur zombie.

Comment les faire sortir ?

Morceaux par morceaux au milieu de requête DNS tout a fait normale

exfiltration DNS

Etape 3 : récupération et exfiltration des plans

Illustration du principe

https://www.root-me.org/fr/Challenges/Forensic/Exfiltration-DNS