Le pharming et la comparaison de pages webs

Sommaire

Au cours des dix dernières années, la prolifération de sites webs contrefaits a conduit à explorer diverses pistes de prévention/détection. La plupart des approches développées s'intéressent aux attaques de phishing - grâce à des techniques d'analyses (p.ex. listes noires, tests heuristiques) appliquées sur l'URL et/ou le contenu de la page web contrefaite (cf. Chapitre 2) -, ou aux attaques de pharming réalisées au travers d'une corruption DNS. Dans le cas de ces dernières, il y a deux manières d'aborder le problème : explorer du côté des informations DNS utilisées pour atteindre le site web, ou investiguer au niveau du contenu de la page web contrefaite.

Dans ce chapitre, nous présentons le pharming (au travers des différentes corruptions DNS qui peuvent être effectuées) et les techniques de comparaisons de pages webs pouvant aider à la détection des sites webs contrefaits. Précisons que les explications délivrées dans ce chapitre sont volontairement simplifiées et limitées, pour ne définir que les notions utilisées ultérieurement dans ce document.

En section 4.1.1, nous débutons par quelques rappels DNS nécessaires à la bonne compréhension des attaques et mécanismes de prévention/détection présentés. Puis en section 4.1.2, nous détaillons les différents niveaux d'attaques de pharming. Enfin, en section 4.1.3, nous évoquons les techniques actuellement disponibles pour s'en prémunir, ainsi que les travaux qui s'y rapportent côté client.

Dans un deuxième temps, nous détaillons les deux grandes catégories de méthodes qui s'avèrent pertinentes pour la comparaison de documents HTML : les méthodes de comparaison de texte (cf. section 4.2.1) et les méthodes de comparaisons de structures (cf. section 4.2.2). Pour terminer, la section 4.2.3 discute plus spécifiquement des travaux qui appliquent ces méthodes au cadre de la comparaison de pages webs.

4.1 Le pharming

Les attaques de pharming sont une version sophistiquée des attaques de phishing. L'objectif est le même, à savoir : voler des informations confidentielles aux Internautes telles que des login, mots de passe, numéros de carte bancaire, etc. D'un point de vue technique, la mise en œuvre de ce type d'attaque diffère quelque peu : il faut certes la mise en ligne d'un site web contrefait mais, basée sur une corruption DNS réalisée en amont, l'attaque est cette fois-ci totalement imperceptible à l'utilisateur (sous réserve que l'imitation visuelle de la page web affichée soit de bonne qualité). En effet, grâce à une corruption/redirection effectuée au niveau DNS, l'utilisateur est automatiquement redirigé vers le site web contrefait dès lors qu'il tente d'accéder à son site habituel. Pourtant l'URL visitée est bien l'URL légitime.

Ce type d'attaque peut être considérée comme plus passive que le phishing, car elle ne nécessite pas de campagne de diffusion incitant les Internautes à se rendre sur le site contrefait. Néanmoins elle n'en est que plus redoutable car difficilement détectable.

Avant de s'intéresser aux attaques de pharming et aux moyens de prévention/détection associés, revenons à quelques rappels/considérations DNS nécessaires à la bonne compréhension de la suite de ce chapitre.

4.1.1 Rappels DNS

Dans le monde de l'Internet, les adresses IP sont utilisées afin d'échanger des messages entre machines. Sachant qu'une adresse IP est constituée de 4 à 8 blocs d'octets séparés par des caractères spéciaux[1], leur manipulation et leur mémorisation n'est pas tâche aisée. Le protocole DNS est donc un élément crucial de l'architecture réseaux d'Internet. A l'image de l'annuaire téléphonique qui permet de lier nom d'abonné et numéro de téléphone, le protocole DNS permet de faire la corrélation entre une adresse IP et le nom de domaine (ou plus exactement le FQDN[2]) associé. Afin de simplifier les échanges au travers d'un réseau, il est en effet plus simple d'utiliser un nom de domaine (p.ex. `www.it-sudparis.eu`) plutôt qu'une adresse IP (p.ex. 157.159.11.8). Par la même occasion, l'administration réseaux qui inclut des besoins de partage de charge ou une migration de serveurs s'en retrouve simplifiée.

Le protocole DNS utilise le port 53, généralement en UDP pour les requêtes et TCP pour les transferts de zone[2]. Il s'appuie sur les RFC fondatrices 1034 (concepts) et 1035 (implémentation). Celles-ci sont complétées par les RFC 2136, 2181, 4033 et 6195[3] qui apportent des spécifications complémentaires sur les enregistrements RR[2] et introduisent les notions de sécurité.

4.1.1.1 Architecture DNS

Le DNS [AFNa] [Oll05] s'appuie sur une architecture client/serveur reposant sur 3 dispositifs :
- Un espace de noms hiérarchique permettant de garantir l'unicité des noms de domaine,
- Un système de serveurs distribués permettant la diffusion de ces noms de domaine,
- Et un système client permettant de résoudre les noms de domaine.

4.1.1.1.1 Espace de noms hiérarchique : La structure hiérarchique des noms de domaine est conçue comme une arborescence, ayant pour origine une racine représentée par un ".", afin que chaque nom d'hôte (c.-à-d. machine) soit unique. L'espace de nommage, c.-à-d. le chemin dans l'arbre inversé pour

1. 4 blocs d'octets séparés par des "." pour les adresses IPv4, et 8 blocs de 2 octets séparés par des " :" pour les adresses IPv6. Notons que l'ensemble de notre étude se focalise exclusivement sur les adresses IPv4.
2. cf. section 4.1.1.1.
3. Liste non exhaustive.

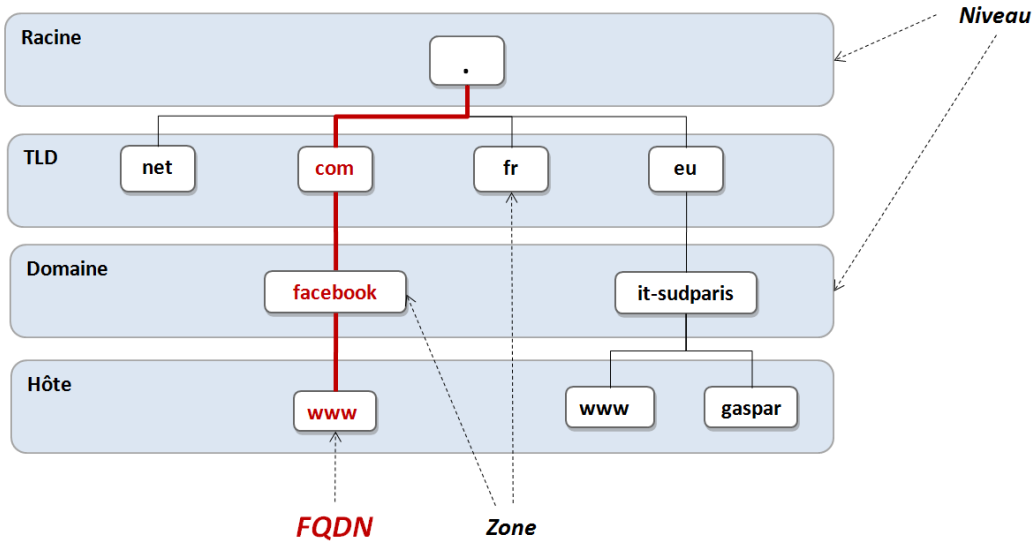

FIGURE 4.1 – Hiérarchie DNS simplifiée

un nom d'hôte donné, comporte typiquement 4 niveaux : racine, TLD, domaine et hôte (cf. figure 4.1). Chaque nom d'hôte (sur 63 caractères maximum) doit être unique dans le domaine concerné. Un domaine peut être éventuellement re-découpé en sous-domaines intermédiaires, tant que le nom d'hôte complet (appelé FQDN et constitué des différents niveaux d'arborescence séparés par des ".") n'excède par 255 caractères et 127 niveaux d'arborescence. Chaque responsable de domaine dispose d'une totale liberté de nommage de ses sous-domaines. Pour un minimum de structure, les TLD - également appelés domaines de premier niveau - suivent une codification spécifique établie par l'ICANN (pour *Internet Corporation for Assigned Names and Numbers*). Ils sont par ailleurs gérés soit directement par l'ICANN, soit par des organismes délégués appelés registres. Enfin, les noms de domaine sont achetés auprès d'un bureau d'enregistrement des domaines, appelé registrar. A noter qu'il est possible de consulter les données contenues par les registres via l'utilisation du protocole WHOIS (RFC 3912, port TCP 43).

Les TLDs se décomposent en plusieurs groupes. Les plus connus sont les g-TLD pour *generic TLD* (p.ex. .COM, .ORG, .GOV, etc.), et les cc-TLD pour *country-code TLD* représentatifs d'un pays ou d'une zone géographique (p.ex. .FR pour la France, .EU pour l'Europe, etc.).

4.1.1.1.2 Système de serveurs distribués : Chaque domaine dispose d'un serveur DNS (au minimum), afin d'annoncer sur Internet les combinaisons FQDN/adresses IPs qu'il détient. Ainsi, il existe des centaines de milliers de serveurs DNS à travers le monde, chacun d'entre eux ne contenant que peu d'informations. A chaque requête DNS effectuée, ce sont en fait de nombreux serveurs DNS qui sont sollicités. On dit de l'architecture DNS qu'elle est un système décentralisé, reposant sur la délégation de domaine. En effet pour que l'ensemble de l'arborescence DNS soit gérable, à chaque niveau (non-terminal) on associe une zone par entité (cf. figure 4.1). Chaque zone est responsable, par délégation de la zone supérieure, de répondre aux requêtes DNS concernant les entités qu'elle détient (c.-à-d. de niveau inférieur).

Traditionnellement, chaque zone contient (jusqu'à) 3 serveurs DNS (cf. figure 4.2) : un serveur primaire contenant les informations de zone (on parle alors de serveur "autoritaire" pour la zone), un serveur de noms secondaire généralement utilisé pour la continuité de services, et un serveur cache qui mémorise les réponses aux précédentes requêtes DNS avec une durée de vie limitée. Les transferts d'informations depuis le serveur DNS primaire vers le serveur DNS secondaire sont appelés transferts de zone.

Le rôle d'un serveur DNS est de répondre à toute requête demandant des informations contenues

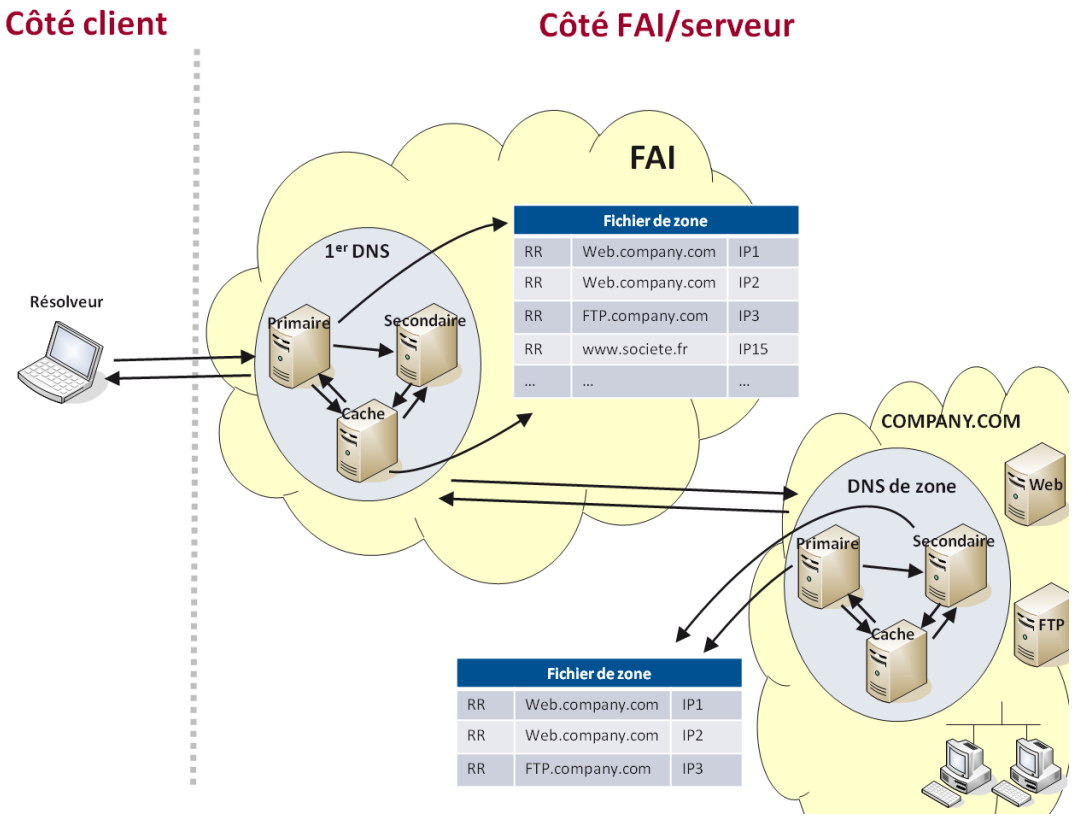

FIGURE 4.2 – Vue simplifiée des zones DNS

TABLEAU 4.1 – Exemples d'enregistrements DNS Resource Record (RR)

RR	FQDN	Type	Classe	TTL	Adresse IP
1	www.google.fr	A	IN	86400	209.85.148.103
2	www.google.fr	A	IN	86400	209.85.148.99
3	www.google.fr	A	IN	86400	209.85.148.105
4	www.it-sudparis.eu	A	IN	43200	157.159.11.8

dans sa base, qu'elles concernent ou non sa zone. Les informations de sa propre zone sont concentrées dans un fichier (nommé fichier de zone) qui contient plusieurs enregistrements appelés *Resource Record* (RR), comme le présente la figure 4.2.

Chaque enregistrement RR contient 5 informations (cf. tableau 4.1) : le FQDN, le Type d'enregistrement, la Classe, le TTL (pour *Time To Live*), ainsi que l'adresse IP associée.

Les types d'enregistrements servent à donner une indication sur la teneur de l'hôte. Les plus connus sont : MX (pour *Mail eXchange*) utilisé pour décrire un serveur mail, NS (pour *Name Server*) utilisé pour décrire le serveur "autoritaire" de la zone, ou A (pour *Address*) utilisé pour décrire une machine/un hôte IPv4 (p.ex. un serveur web, un serveur ftp, etc.). Le TTL permet, quant à lui, d'associer une durée de vie (exprimée en secondes) à chaque enregistrement. Enfin, la Classe permet de décrire le système associé à l'enregistrement. On utilise ici IN pour indiquer *Internet*.

4.1.1.1.3 Système client : Le système client DNS, permettant de réaliser l'opération de résolution de domaine, est appelé résolveur (ou *resolver* en anglais). Installé côté client, il est appelé par les applications du système en vue d'effectuer les requêtes DNS. Ainsi, par l'intermédiaire du résolveur, une application souhaitant contacter un hôte par son nom de domaine envoie une requête au serveur DNS dont il détient l'adresse (typiquement celui du FAI, automatiquement configuré via DHCP, dans le cas d'un réseau personnel).

Trois modes de fonctionnement se présentent alors :

1. Le serveur DNS interrogé possède les informations dans son serveur cache, il répond directement à la requête.
2. Le serveur DNS interrogé n'a pas les informations en cache, mais il connaît un des serveurs de zone du domaine demandé. Il interroge alors directement ce serveur pour obtenir l'adresse IP demandée et ainsi répondre à la requête.
3. Le serveur n'a ni les informations en cache, ni connaissance de la zone demandée. Il génère alors une requête DNS vers un serveur racine pour obtenir une réponse.

A noter qu'un client peut également effectuer une requête DNS à partir d'une adresse IP, en vue de connaître le FQDN associé. On parle alors de requête inverse ou *reverse DNS*.

4.1.1.2 Processus de résolution des requêtes DNS

Les requêtes DNS peuvent être de 2 types : récursives ou itératives. Dans le cas d'une requête récursive, le serveur DNS interrogé doit obligatoirement répondre à la requête (avec une adresse IP/FQDN ou un message d'erreur). Il s'agit là du cas typique de requête générée par un résolveur.

Dans le cas d'une requête itérative, le serveur DNS interrogé retourne tous les éléments en sa possession au demandeur, afin que celui-ci se charge lui-même de poursuivre les requêtes vers d'autres serveurs DNS. Ce type de requête est typiquement utilisée (sauf demande contraire) entre serveurs DNS. Elle est privilégiée ici afin de réduire la charge des serveurs DNS.

4.1.1.3 Exploitation du DNS

De par la quantité colossale de machines qui nécessitent des enregistrements DNS (p.ex. les serveurs webs, emails, FTP, etc.) et la quantité encore plus importante de demandeurs potentiels (c.-à-d. toute machine disposant d'une adresse IP peut émettre des requêtes DNS), l'architecture DNS constitue l'un des socles d'Internet. A ce titre, elle se doit de répondre à des exigences considérables :

- Un service ininterrompu, c.-à-d. les informations DNS doivent être disponibles à tout instant, depuis n'importe quel coin du globe et ce, même en cas de panne d'un serveur DNS.
- Une rapidité pour la réponse au demandeur, indépendamment de la distance géographique entre le client et le service recherché.
- Une répartition de charge optimisée pour ne saturer ni les serveurs DNS interrogés, ni les services accédés.

Pour parvenir à satisfaire ces exigences, plusieurs mécanismes sont mis en œuvre au sein des réseaux des FAI et des entreprises :

- L'utilisation d'un trio de serveurs DNS (c.-à-d. serveur primaire, serveur secondaire et serveur de cache), tant pour assurer la continuité de services que la rapidité des réponses.
- L'utilisation de techniques de répartition de charge pour ne pas saturer les serveurs ciblés. En effet, dès lors que plusieurs adresses IP sont associées à un même hôte au sein des enregistrements DNS contenus dans sa base, le serveur DNS qui répond permute l'ordre des adresses IP retournées, de manière automatique à chaque requête. Cette permutation est réalisée grâce à des techniques telles que le *round-robin* ou le GSLB (pour *Global Server Load Balancing*).

Ces mécanismes présentent toutefois des inconvénients. En effet, l'utilisation de plusieurs serveurs DNS (c.-à-d. primaire et secondaire) entraîne des besoins de réplication de bases et mises à jour, réalisés via les transferts de zones. Ces derniers sont des cibles supplémentaires pour les attaques DNS [Gue06].

Par ailleurs, les techniques de répartition de charge augmentent la quantité d'informations présentes dans les caches DNS. De plus, le *round-robin* effectue une répartition "basique" qui ne tient pas compte de la disponibilité de service associée à chaque adresse IP contenue dans les enregistrements DNS d'un hôte. Les solutions de type GSLB [Net09], qui nécessitent la duplication des services sur un minimum de 2 zones distinctes, permettent de pallier ce problème via une répartition de charge DNS "intelligente", effectuée en fonction de la localisation de l'utilisateur et de la disponibilité du service demandé (c.-à-d.

la capacité des serveurs et leur disponibilité/charge). Cependant, le développement de l'utilisation de ces techniques rend difficilement exploitable le recours aux enregistrements DNS pour identifier d'un domaine (cf. section 5.2.1.4).

4.1.2 Les attaques de pharming

Les attaques ciblant le DNS, typiquement vectrices de pharming, visent à altérer le processus normal de résolution des requêtes DNS. Elles sont classifiées en sous-ensembles qui diffèrent selon les articles et études considérés. Pour notre part, nous avons choisi de les scinder en deux grandes familles, représentatives de la zone attaquée : côté client ou côté réseau FAI/serveur web (cf. figure 4.3).

A noter que dans cette section nous mentionnons également quelques attaques visant le serveur web visité, afin d'offrir une meilleure vue d'ensemble de la chaîne de liaison client - serveur web, ciblée par le vol d'informations confidentielles.

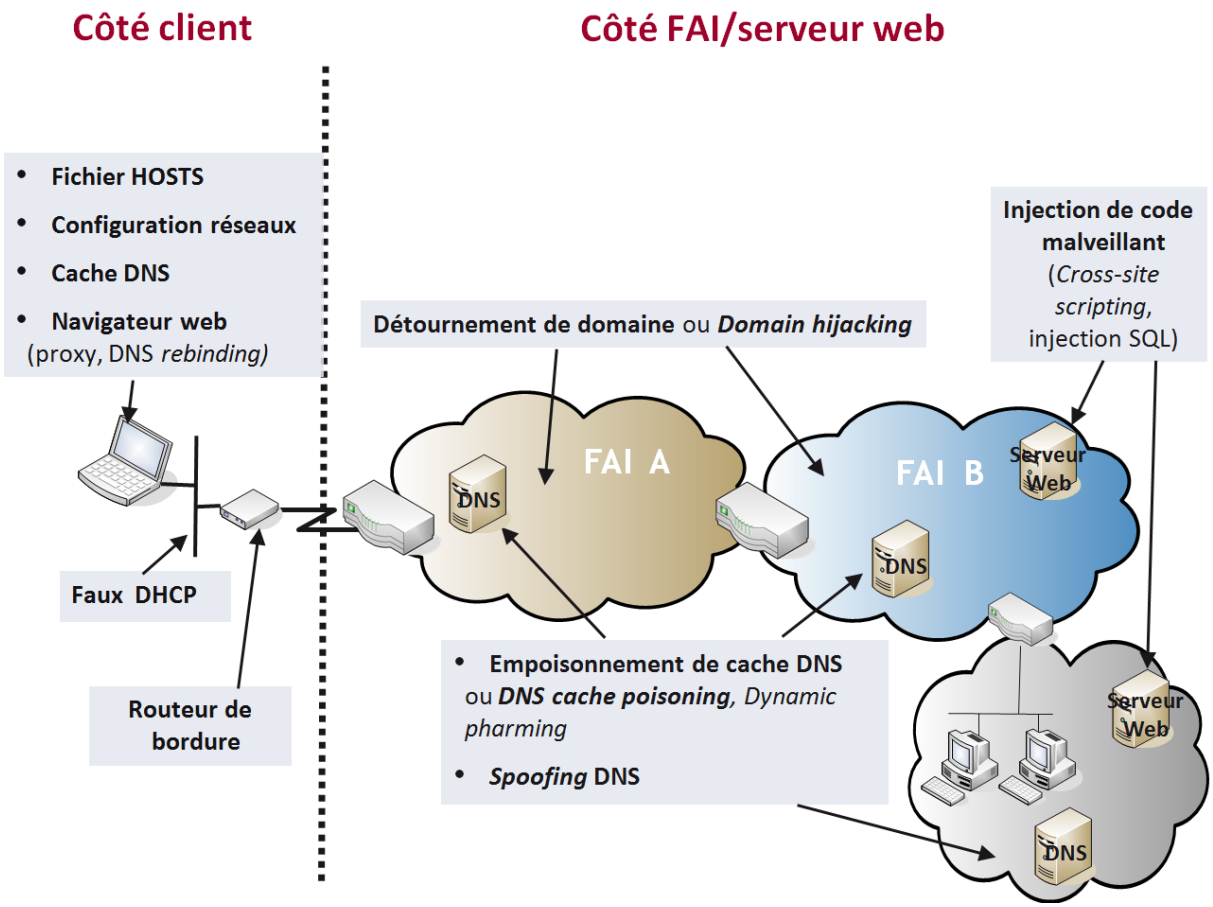

FIGURE 4.3 – Les attaques de pharming réparties par zones cibles

4.1.2.1 Côté client

Les attaques DNS perpétrées côté client [Oll05] peuvent être conduites à 3 niveaux :
- Directement sur le poste client : au niveau du système d'exploitation ou au niveau du navigateur web,
- Sur le réseau local,
- Ou à la frontière entre le réseau local et le FAI, c.-à-d. sur le routeur personnel/de bordure.

Au niveau du poste client, on recense 4 zones de vulnérabilités et/ou cibles d'attaques :

Une première cible peut être le **fichier HOSTS**, présent dans tout système d'exploitation (p.ex. accessible sous Windows dans le répertoire C :\windows\system32\drivers\etc\). Ce fichier sert en effet à définir des associations adresses IP / FQDN de manière statique. Il est consulté par le système, en amont de toute requête DNS envoyée à un serveur.

Une deuxième cible peut être la **configuration réseaux** paramétrée au niveau de la carte d'interfaces. En effet, il est possible de spécifier de manière statique les adresses IP des serveurs DNS interrogés (c.-à-d. primaires et secondaires) au niveau de la carte réseaux. Ces paramètres remplacent alors ceux traditionnellement assignés lors de la configuration dynamique réalisée par DHCP. Dans le cadre d'une utilisation normale, cette zone de configuration est modifiée par les utilisateurs qui souhaitent utiliser un serveur DNS alternatif à celui proposé par leur FAI.

Une troisième cible est le **cache DNS** du poste client, manipulé soit au niveau du système d'exploitation, soit au niveau du navigateur web. Par insertion de fausses entrées (adresse IP/FQDN) dans le cache, qui est consulté avant chaque émission d'une requête DNS, l'utilisateur est alors automatiquement redirigé sur le(s) site(s) web(s) frauduleux.

Ces trois premières cibles peuvent être typiquement corrompues par des scripts malveillants, récupérés au travers d'un spam ou de la navigation web de l'Internaute. Ces scripts se basent aussi bien sur le manque de vigilance/la crédulité des utilisateurs (p.ex. *Wareout* [Mic08] - récupéré par un simple clic lors de la navigation web -, ou le cheval de troie Zcodec - caché à l'intérieur d'un lecteur vidéo dit gratuit [DPLL08] -, configurent statiquement des adresses DNS pirates), que l'exploitation des vulnérabilités du système client (p.ex. une vulnérabilité Microsoft [Mic09] permettait d'ajouter de fausses entrées DNS dans le cache).

Une quatrième zone de vulnérabilités est le **navigateur web** du client. Il peut être corrompu au niveau du cache, comme énoncé précédemment.

Sinon, un **proxy web** peut également être défini au sein du navigateur afin d'entraîner une redirection automatique de toutes les requêtes HTTP vers un serveur web frauduleux.

Enfin, le navigateur web peut être ciblé par une attaque de type ***DNS rebinding*** qui vise à le convertir en proxy ouvert, via l'exploitation des interactions entre plug-ins (p.ex. Flash, Java) et navigateur web [JBB+09]. L'objectif final de cette attaque est d'atteindre et corrompre une machine du réseau local (p.ex. pour ajouter de fausses entrées DNS dans un PC ou le routeur personnel).

Le réseau local, filaire ou sans-fil est également une zone de vulnérabilités. Par l'installation d'un **faux** serveur **DHCP** (p.ex. le cheval de troie Flush.M [Sym08]), un attaquant peut alors assigner de faux paramètres DNS aux utilisateurs. Cette vulnérabilité est d'ailleurs exacerbée par la forte utilisation des réseaux sans-fils déployés à domicile, pour l'accès à Internet. L'étude menée par Abu-Nimeh et al. est un exemple flagrant de la mise en œuvre de ce scénario [AN08].

Enfin, le **routeur personnel** situé à la frontière avec le réseau Internet - traditionnellement fourni par le FAI -, est lui aussi une cible privilégiée des attaquants. Une étude menée par Stamm et al. [SRM07] a démontré la vulnérabilité de ces routeurs personnels, basée sur la double exploitation des lacunes de configuration de ces routeurs et du support du Javascript et des applets Java dans le navigateur client.

En effet, bon nombre d'utilisateurs utilisent ces boîtiers prêts-à-installer dans leur pré-configuration d'origine, sans autre paramétrage de sécurité que la définition d'un mot de passe sécurisant la connexion sans-fil PC(s) - routeur. Les utilisateurs omettent alors généralement de modifier les paramètres d'accès à la configuration du routeur, laissant ainsi activés les login/mot de passe par défaut. En quelques clics sur Internet, il est rapide et aisé de trouver des listes de paramètres d'accès par défaut (c.-à-d. login/mot de passe) à ces équipements [Phe10]. Après une identification de l'adresse IP utilisée par le routeur (p.ex. grâce à une écoute discrète effectuée à partir d'applets Java), et l'utilisation de requêtes HTTP (intégrées à l'intérieur de balises <script> dans le code source d'une page web), l'accès à la configuration DNS du routeur personnel peut être réalisé. On peut alors spécifier - de manière statique - l'adresse IP d'un faux serveur DNS, ou créer des règles de routage qui redirigent le trafic.

Bien évidemment, les cas d'attaques exposés dans cette étude répondent à des conditions spécifiques (p.ex. la nécessité d'un mot de passe par défaut vide pour certaines attaques, l'acceptation de messages d'alertes par l'utilisateur pour d'autres, etc.). Néanmoins, elles sont tout à fait viables.

Il est important de souligner que cet article [SRM07], que nous avons considéré comme marquant, a été l'un des points d'origine de notre réflexion dans la recherche d'une méthode de détection des attaques de pharming côté client.

4.1.2.2 Côté réseau FAI/serveur web

Côté réseau FAI/serveur web, les attaques peuvent se situer à différents niveaux [Oll05] :

- Détournement du nom de domaine (ou *Domain hijacking* en anglais) par modification de son enregistrement,
- Altération du fonctionnement normal d'un serveur DNS par *spoofing* DNS,
- Corruption de cache DNS (ou *DNS cache poisoning* en anglais) au niveau d'un serveur DNS,
- Ou directement au niveau du serveur web visité, par l'injection de code malveillant.

Le **détournement du nom de domaine** (ou *Domain hijacking* en anglais) consiste à s'approprier un nom de domaine légitime, basé sur des failles liées au processus d'enregistrement de celui-ci. Le cas du domaine `Panix.com`, détenu par un FAI qui délivre des accès Internet sur New-York, en est un exemple. En effet, à cause d'un oubli dans le processus d'enregistrement, le domaine du FAI a été détourné (c.-à-d. les adresses IP ont été modifiées) au profit d'une société Australienne, durant quelques dizaines d'heures en 2005 [spe05]. Ce type d'attaque, relativement rare, ne fonctionne généralement que sur une courte durée.

Le ***spoofing* DNS** consiste à altérer le fonctionnement normal d'un serveur DNS légitime afin d'émettre de fausses réponses DNS. Ces dernières sont alors délivrées soit par le serveur DNS légitime lui-même (p.ex. grâce à l'utilisation de techniques d'empoisonnement de cache), soit par le serveur DNS de l'attaquant, en lieu et place du serveur légitime (p.ex. via des techniques de dénis de service conduisant à un arrêt ou un ralentissement des réponses du serveur DNS légitime). Ces attaques peuvent cibler le serveur DNS d'un FAI, ou le serveur DNS autoritaire d'une zone (p.ex. par corruption des transferts de zone).

Les attaques de type ***DNS cache poisoning*** visent à corrompre le cache d'un serveur DNS (localisé chez un FAI ou dans une zone) par insertion de faux couples d'adresses IP/FQDN. Elles sont souvent utilisées en association avec une attaque de pharming basée sur du *spoofing* DNS.

Une variante de ce type d'attaque - appelée *Dynamic pharming* - combine la délivrance au client d'un code Javascript malveillant et l'utilisation de techniques de *DNS rebinding* (cf. section 4.1.2.1). L'association de ces deux techniques permet d'incorporer la page web légitime au sein de la page web malveillante visualisée par le client.

La figure 4.4, basée sur celle contenue dans l'article de Olzak [Olz06], montre un exemple d'attaque de pharming combinant des techniques de *DNS cache poisoning* pour corrompre le cache DNS du FAI (via l'étape 7b de la figure), et de *spoofing* DNS pour effectuer un déni de service sur le serveur DNS de la zone *company.com*.

Enfin, nous pouvons mentionner les **injections de code malveillant** (par *Cross Site Scripting* (XSS) ou injection de code SQL) qui ciblent le serveur web visité.

Ces techniques consistent à insérer des scripts malveillants et/ou des redirections de contenu au sein d'une page légitime (p.ex. insertion d'une fausse zone de saisie de login/mot de passe), via l'exploitation de failles inhérentes aux serveurs et/ou navigateurs webs utilisés. A titre d'exemple, on peut citer le cheval de troie Sinowal [RSA08] qui était exécuté sur le poste client. Actif sur plus de 2700 URLs de services financiers, il a permis de dérober les données de plus de 300 000 comptes bancaires sur une période de 3 ans.

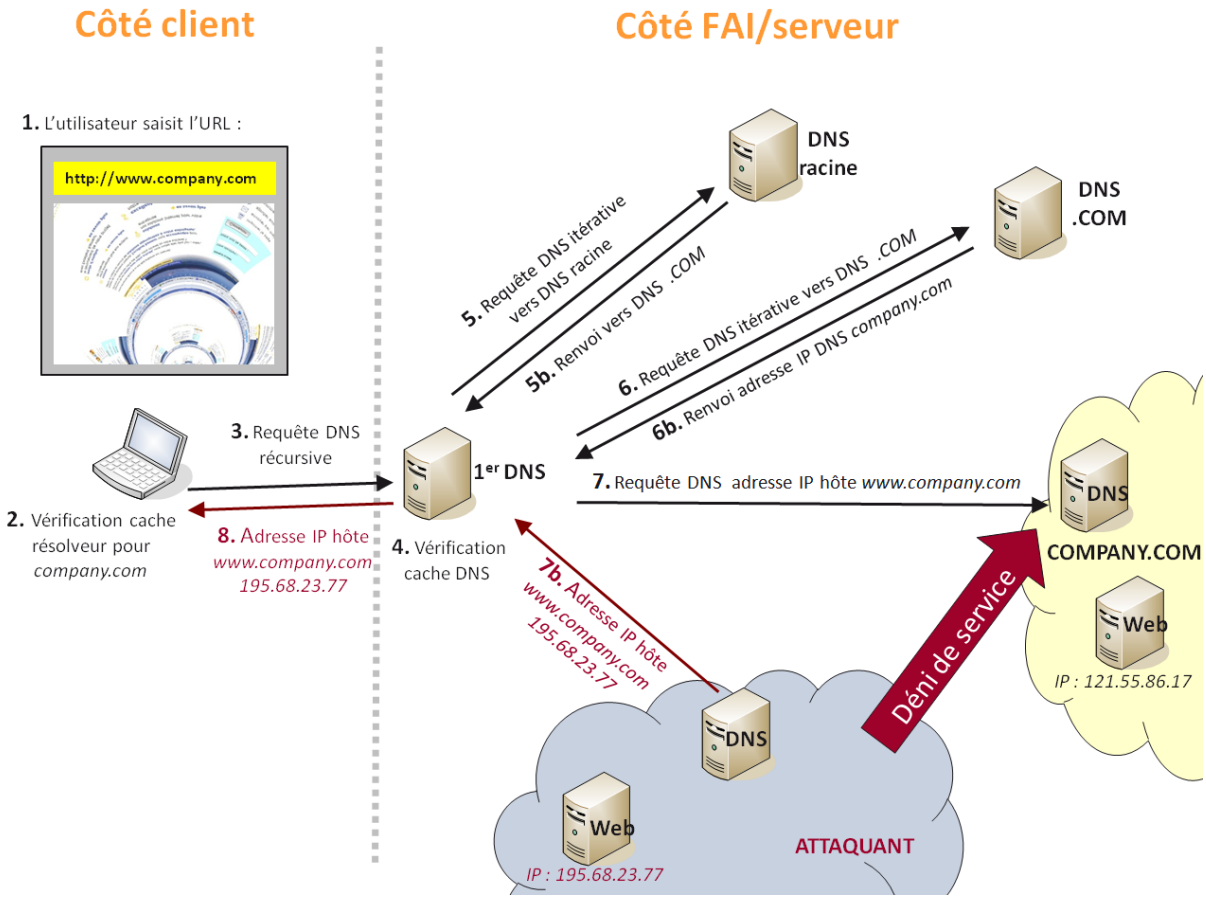

FIGURE 4.4 – Exemple d'attaque de pharming combinant des techniques de *DNS cache poisoning* et de *spoofing* DNS

4.1.3 Méthodes de prévention/détection du pharming

Les attaques de pharming s'appuient sur les faiblesses du protocole DNS et/ou les zones de configuration associées, qu'elles se situent côté client ou côté réseau FAI/serveur web.

A son origine - à l'image des protocoles associés à IPv4 -, le protocole DNS a été conçu sans aucune notion de sécurisation. En effet, durant les années 80, la priorité était au développement de protocoles efficaces et performants pouvant répondre aux besoins de croissance des réseaux [AFN10].

Depuis, de par le développement de l'utilisation du DNS - désormais inéluctable aux échanges sur Internet - ainsi que les nombreuses zones de vulnérabilités associées, diverses solutions de sécurité visant à améliorer le DNS et/ou protéger des attaques de pharming ont été proposées.

4.1.3.1 DNSSEC : une solution côté réseau FAI/serveur web

La proposition majeure associée à la sécurité du DNS concerne le développement des extensions de sécurité DNSSEC (pour *Domain Name System SECurity*) dont les bases fondatrices sont définies dans les RFC 4033, 4034 et 4035[1].

4.1.3.1.1 Fonctionnement général : DNSSEC [AFNb] [Gue06] vise à sécuriser l'architecture de nommage DNS dans sa globalité, tant au niveau des échanges effectués entre serveurs DNS, que des différents niveaux d'arborescence non-terminaux (cf. figure 4.1).

Ces extensions de sécurité reposent sur des mécanismes de signatures (générées à l'aide de chiffrement asymétrique) pour assurer l'authenticité et l'intégrité (c.-à-d. les informations n'ont pas été

1. Liste non exhaustive.

Tableau 4.2 – Exemples d'enregistrements DNS
Resource Record set (RRset)

RRset	FQDN	Type	Classe	TTL	Adresse IP
1	www.google.fr	A	IN	86400	209.85.148.103
		A	IN	86400	209.85.148.99
		A	IN	86400	209.85.148.105
2	www.it-sudparis.eu	A	IN	43200	157.159.11.8

altérées durant leur transfert) des enregistrements DNS [AFN10].

Supposons le cas d'un client A (ou plus exactement son résolveur) qui souhaite connaître l'adresse IP associée au serveur web B, afin de répondre à la demande d'une application cliente. Le mode de fonctionnement à minima des échanges DNS en utilisant DNSSEC vise à être le suivant :

1. Au préalable, le serveur DNS autoritaire de la zone B a publié sa clé publique.
2. Au préalable, le serveur DNS autoritaire de la zone B a publié ses informations DNS (c.-à-d. ses couples d'adresses IP/FQDN) accompagnées d'une signature. Cette signature a été générée localement à partir de la clé privée du serveur DNS B et des informations à publier.
3. En retour de la requête DNS émise par le résolveur du client, une réponse DNS est reçue. Celle-ci contient soit l'information demandée (c.-à-d. un couple d'adresses IP/FQDN) accompagnée de sa signature, soit un message d'erreur.
4. Si la réponse DNS contient l'information demandée : la machine DNSSEC validante côté client (c.-à-d. la dernière machine qui implémente DNSSEC : soit le serveur DNS du FAI auquel est rattaché le client, soit le résolveur demandeur) se charge alors de vérifier que l'association adresses IP/FQDN retournée a bien été émise par le serveur DNS B[1]. En effet, par l'utilisation de la clé publique publiée par B (et diffusée dans l'arborescence DNS), la signature accompagnant les informations est vérifiée.
5. Si la réponse DNS contient un message d'erreur : la machine DNSSEC validante côté client peut vérifier que ce message a bien été émis par le serveur DNS B (via la vérification de la signature). Si c'est le cas, il peut en supplément s'assurer que le domaine interrogé référence ce type d'enregistrement.

A noter que, comme précédemment (c.-à-d. avec l'utilisation du DNS seul, sans extensions de sécurité), les informations DNS d'une zone sont diffusées à l'ensemble des serveurs DNS d'Internet, au travers des échanges réalisés entre serveurs DNS parents.

Avec l'apport de DNSSEC, les échanges d'informations DNS entre ces serveurs DNS parents se font désormais de manière sécurisée.

En supplément, afin d'éviter la création/utilisation de fausses clés DNSSEC par un attaquant, un serveur DNS parent signe (à l'aide de sa propre clé privée) chacune des clés publiques des zones qui lui sont rattachées (selon le même procédé qu'expliqué précédemment). Par conséquent, pour une sécurité optimale, chaque machine validante DNSSEC recevant des enregistrements DNS se doit de vérifier leur appartenance à la zone annoncée, via l'utilisation de la clé publique du serveur DNS parent associé.

4.1.3.1.2 Précisions techniques : L'implémentation de ces extensions de sécurité DNSSEC entraîne l'apparition de notions/enregistrements DNS additionnels [Gue06]. Parmi les plus connus, on peut citer notamment les enregistrements RRset, RRSIG, DNSKEY, DS, NSEC (ou NSEC3) et les clés KSK, ZSK :

Pour limiter la quantité d'informations nouvellement stockées par l'apport de DNSSEC, les enregistrements RR de même type, même nom d'hôte et même classe sont regroupés au sein d'un ensemble d'enregistrements nommé RRset (pour *Resource Record set*, cf. tableau 4.2).

Ainsi le serveur DNS autoritaire d'une zone signe un RRset, en lieu et place de signer chaque RR. Les signatures résultantes sont alors stockées dans des enregistrements nommés RRSIG (pour *Resource*

1. Cette vérification est effectuée avant transmission de la réponse au résolveur, si la dernière machine DNSSEC validante est le DNS du FAI.

Record SIGnature). Il y a autant d'enregistrements RRSIG que de RRset signés avec une clé privée donnée. Les signatures sont générées à partir d'un algorithme de chiffrement asymétrique de type RSA ou DSA, combiné à une fonction de hachage de type MD5, SHA.

La clé publique du serveur DNS autoritaire d'une zone est nommée ZSK (pour *Zone Signing Key*). Elle est publiée, dans le serveur DNS autoritaire de la zone, via un enregistrement DNSKEY (pour *Domain Name System KEY*). Précisons qu'un serveur DNS autoritaire de zone peut utiliser plusieurs couples de clés privées/clés publiques. Dans ce cas, chaque clé publique fait l'objet d'un enregistrement DNSKEY et chaque RRset doit être signé par chacune des clés privées de la zone.

Le(s) enregistrement(s) DNSKEY sont également référencés dans le serveur DNS de la zone parente, via un(des) enregistrement(s) DS (pour *Delegation Signer*). Le serveur DNS de la zone parente signe les enregistrements DNSKEY de ses zones filles avec une clé nommée KSK (pour *Key Signing Key*).

Enfin, dans le cas où la requête DNS est un succès, le demandeur reçoit le RRset de l'hôte recherché, accompagné de la signature RRSIG associée. Dans le cas idéal, la machine validante DNSSEC (c.-à-d. le demandeur ou le serveur DNS du FAI) effectue une vérification en 3 étapes : 1/ elle récupère la clé DNSKEY de la zone émettrice, 2/ elle s'assure de la légitimité de la DNSKEY récupérée en contactant le serveur DNS parent de la zone, et 3/ si DNSKEY est légitime, elle l'utilise afin de vérifier l'authenticité et l'intégrité du RRset reçu.

A contrario, en cas d'échec de la requête DNS, le demandeur reçoit un enregistrement NSEC ou NSEC3 (pour *Next-SECure record*) en guise de message d'erreur. Contrairement au simple code d'erreur reçu avec l'utilisation du DNS sans extension de sécurité, le message NSEC ou NSEC3 donne des indications sur la légitimité du répondant et l'existence de l'hôte demandé.

La figure 4.5 montre une vue d'ensemble des liens entre enregistrements DNSSEC (en pointillés sur le schéma), ainsi qu'un exemple de requête DNS/DNSSEC (en traits continus sur le schéma).

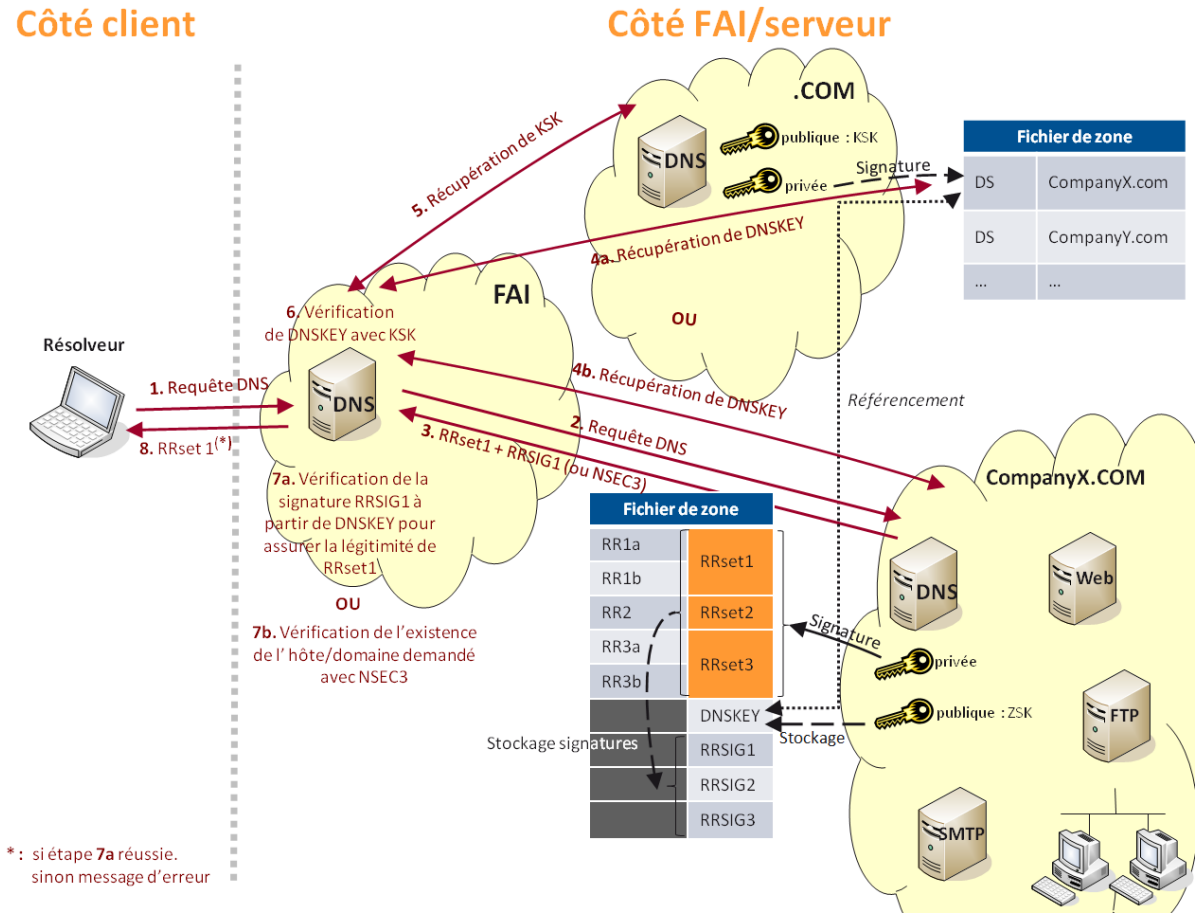

FIGURE 4.5 – Exemple de requête DNS avec extensions de sécurité DNSSEC

4.1.3.1.3 Point sur le déploiement DNSSEC : La vulnérabilité découverte par Dan Kaminsky [Sch08] durant l'été 2008, a été le point d'orgue au déploiement des extensions de sécurité DNSSEC, aux plus hauts niveaux de l'arborescence DNS. La faille découverte - assimilée à de l'empoisonnement de cache DNS - a révélé à tous le caractère d'urgence associé à la mise en œuvre de DNSSEC. Depuis, les serveurs TLD ont donc progressivement commencé à déployer ces extensions de sécurité. Parmi les plus connus, on peut notamment citer les TLD .ORG, .GOV, .UK, .EDU, .FR, .COM et .NET.

Au 15 Juillet 2010, une avancée majeure a été réalisée : le déploiement de DNSSEC a été finalisé au niveau des 13 serveurs racines [IV10].

Néanmoins, malgré ces efforts, l'ensemble de l'arborescence DNS est loin d'avoir migré. Il faut en effet que l'ensemble des acteurs d'Internet (c.-à-d. FAI, bureaux d'enregistrements, entreprises, clients, etc.) s'y mettent à leur tour.

Côté client/entreprise, Microsoft a par ailleurs annoncé la mise en œuvre de fonctionnalités DNSSEC dans les systèmes d'exploitation Windows 7 et Windows Server 2008 R2 [Mic10]. Toutefois, la mise en œuvre des extensions de sécurité côté client passe d'abord par la mise en œuvre côté réseau.

Par conséquent, aujourd'hui on ne peut que trop rarement s'attendre à voir DNSSEC déployé jusqu'au résolveur du particulier. A défaut, la vérification des enregistrements DNSSEC se voit donc assurée par le serveur DNS le plus proche, à savoir : celui du FAI.

4.1.3.1.4 Limitations : Il est important de préciser ce que DNSSEC ne fait pas [AFN10] : on peut notamment citer qu'il n'assure pas la confidentialité des enregistrements DNS, puisque ceux-ci sont uniquement signés. Il ne vise pas non plus à se substituer aux mécanismes d'authentification actuellement utilisés pour sécuriser les transactions électroniques (p.ex. via HTTPS). Il ne peut rien contre le phishing, une grande partie des attaques effectuées côté client ou les attaques réalisées sur le serveur web. Il ne peut non plus garantir d'une éventuelle compromission des enregistrements DNS en amont de leur insertion dans le serveur DNS autoritaire.

De plus, au travers de l'ajout d'extensions qui visent à introduire davantage de sécurité (p.ex. la délivrance d'un message d'erreur consistant et explicite qui permet d'appréhender l'existence de l'hôte demandé), DNSSEC a également introduit d'autres vulnérabilités et/ou problèmes [BM10].

On peut notamment citer le cas de l'enregistrement NSEC. Celui-ci peut permettre à un attaquant de prendre connaissance des hôtes du domaine légitime. Pour contrer ce problème, un nouvel enregistrement NSEC3 a donc été développé, afin de masquer les noms des hôtes.

On peut également parler des problèmes associés au chiffrement introduit, p.ex. la lourdeur de gestion des clés dans une arborescence aussi conséquente (c.-à-d. assurer une cohérence dans les clés utilisées, gérer leur renouvellement), la bonne gestion des interactions serveur autoritaire - serveur parent lors des mises à jour de clés (de par la forte relation entre les enregistrements DS et DNSKEY), ou encore l'absolue nécessité de diffuser rapidement les mises à jour de clés/signatures dans l'arborescence DNS.

4.1.3.2 Le réseau client reste une cible privilégiée

Auparavant les principales techniques de sécurisation associées au DNS résidaient davantage en des techniques de supervision (p.ex. remontées d'alertes spécifiques, mises à jour logicielles) et des mesures de continuité de service [AFN09]. L'ensemble des zones de vulnérabilités, et attaques associées, exposées en section 4.1.2 démontre bien les lacunes sécuritaires du protocole DNS.

De par la structure DNS, il apparaît que la compromission des informations DNS côté client est certainement moins visible que côté réseau. En effet, les serveurs DNS des FAI et des plus hauts niveaux d'arborescence sont sous haute surveillance. Une attaque sur le service recherché (c.-à-d. le serveur web et/ou son domaine) peut alors être préférée mais, là encore, elle deviendra rapidement visible. Par conséquent, le réseau client est donc une cible privilégiée, même si plus contraignante car la compromission à large échelle doit être effectuée en une multitude de points.

L'apport de DNSSEC est donc bien une nécessité incontournable visant à éradiquer/minimiser les compromissions des enregistrements DNS au sein d'Internet. Toutefois DNSSEC ne deviendra réellement efficace que lorsque l'ensemble de la chaîne de liaison DNS (du client jusqu'au serveur web visité) l'aura déployé. De plus, il est indispensable que les informations DNSSEC soient correctement

publiées/renouvelées par les serveurs autoritaires, et que les mises à jour soient rapidement diffusées à l'ensemble de la chaîne de liaison. Dans le cas contraire, des indisponibilités des services recherchés sont à prévoir.

Côté client, tant que le résolveur n'aura pas mis en œuvre DNSSEC, aucune protection supplémentaire ne sera apportée si ce n'est la garantie que les données DNS n'auront pas été corrompues entre le DNS du FAI et celui de la zone du serveur web visité. Par ailleurs, même en cas de déploiement sur le résolveur client, DNSSEC ne protège pas de l'ensemble des attaques perpétrées côte client (p.ex. les ajouts de fausses entrées IP/FQDN statiques sur le poste client, telles qu'exposées en section 4.1.2.1).

Enfin, précisons également que l'utilisation du protocole HTTPS est insuffisante pour se prémunir des attaques de pharming. En effet, l'attaquant peut également corrompre les paramétrages de sécurité effectués au niveau du poste client, pour altérer la vérification et/ou l'alerte faite à l'utilisateur en cas de certificat non valide. Certaines propositions exposées en section 4.1.3.3 s'intéressent d'ailleurs à cette problématique.

4.1.3.3 Des propositions côté client

Les propositions qui ciblent le côté client indiquent plusieurs vecteurs de sécurisation des informations DNS et/ou de protection envers les sites contrefaits. Elles sont étayées par deux types d'approches : des techniques amont qui visent à restreindre le champs d'accès des attaquants, ou des techniques de détection utilisées lors de la navigation web. Ces dernières peuvent aussi bien travailler sur les informations de connexion (adresse IP, certificat SSL, etc.) que sur le contenu de la page web, à la recherche de comportements anormaux.

4.1.3.3.1 Restreindre le champ d'accès des attaquants :

Poste client : De manière générale, l'ensemble des propositions qui s'intéressent au poste client visent à mieux contrôler le contenu des pages webs visitées (c.-à-d. mieux protéger le navigateur web) ou les vecteurs qui y conduisent. En effet, de par la forte exploitation des applets Java (pour obtenir les adresses IP du client) et des scripts malveillants (incorporés au code HTML des pages via des scripts Javascript) pour diffuser les compromissions DNS côté client, les études se rejoignent sur le besoin de mieux contrôler l'accès et l'exécution de ces contenus. Pour exemple, Barth et al. [BFSB10] ont démontré que 88% des 25 extensions couramment utilisées par les utilisateurs de Firefox et testées dans leur étude, donnent davantage de points d'accès à la configuration du système qu'elles n'en nécessitent pour fonctionner. Par conséquent, elles introduisent autant de brèches potentielles pour une éventuelle compromission par un attaquant. Les auteurs recommandent donc de revoir la plateforme utilisée pour la création des extensions en accordant une meilleure place à la sécurité (c.-à-d. en n'accordant à une extension que les droits d'accès strictement nécessaires à son bon fonctionnement).

Pour contrer et/ou détecter les corruptions DNS effectuées côté client, Ollmann [Oll05] propose de désactiver les fonctionnalités qui autorisent l'affichage de contenu dynamique et/ou personnalisé (p.ex. ActiveX, fenêtres pop-up, cookies, etc.). Plus spécifiquement, Stamm et al. [SRM07] recommandent de restreindre l'accès aux fichiers de configuration système (c.-à-d. qui gouvernent la configuration IP/DNS) aux applets qui sont signées. Dans la même lignée, des extensions de sécurité pour navigateurs sont désormais disponibles afin de mieux contrôler les exécutions automatiques de scripts incorporés dans les pages webs. On peut par exemple citer des extensions comme NoScript [Inf] et Sabre [DG09], disponibles pour Firefox, qui aident à mieux contrôler les Javascripts exécutés au travers de listes blanches.

Une autre proposition de Stamm et al. [SRM07] réside en l'exclusion des scripts, issus d'un site web tiers (p.ex. des publicités), dans l'affichage de la page web visitée par l'utilisateur. L'étude de Karlof et al. [KSTW07] qui porte sur les connexions SSL, rejoint cette idée. Elle propose le renforcement des SOP (pour *Same Origin Policy*) par l'insertion d'indicateurs, révélateurs de la cohérence des informations de sécurité obtenues lors de l'établissement de la connexion sécurisée. Les SOP, généralement utilisés par les codes Javascript, permettent d'autoriser sans restriction l'exécution de scripts issus d'un domaine donné au sein d'une page web de même origine (c.-à-d. de même domaine). Ainsi dans la proposition évoquée, un script ne pourra être exécuté sans restriction que s'il y a concordance du nom de domaine et

de l'indicateur inséré. Ce dernier peut refléter, par exemple, une incohérence entre les noms de domaine présents dans l'URL et le champ CN (pour *Common Name*) du certificat SSL délivré par le serveur web.

Réseau client : De manière globale, les études considérées ([JBB+09], [SRM07] et [Oll05]) préconisent d'utiliser des outils de diagnostic réseau disponibles en ligne ou installés dans le réseau client, afin de détecter d'éventuelles compromissions (p.ex. des outils de scan pour détecter d'éventuels malwares installés sur le poste client, analyser régulièrement les paramètres DNS, etc.).

Routeur de bordure : L'étude de Stamm et al. [SRM07] propose d'abandonner les mots de passe par défaut actuellement pré-configurés (p.ex. *(vide)*, admin, etc.) pour le compte administrateur qui sert à la configuration IP/DNS du routeur. Les auteurs proposent de le remplacer, par exemple, par le numéro de série de l'équipement afin de rendre son identification moins facile.

4.1.3.3.2 Détecter l'attaque lors de la navigation web :

Des extensions de sécurité pour navigateurs peuvent être utilisées pour valider les informations de sécurité reçues par Internet, lors de la navigation web. On peut par exemple citer Perspectives [WAP08] pour Firefox, ou HTTPSLock [FC10] qui visent à vérifier les certificats SSL auprès d'une liste blanche (p.ex. via un réseau notaire).

Cao et al. [CHL08] proposent, quant à eux, une méthode de détection des sites de pharming/phishing basée sur une comparaison de l'adresse IP du domaine visité auprès d'une liste blanche d'adresses IP légitimes. Cette méthode s'appuie sur un principe de staticité des adresses IP associées aux pages de login.

Basés sur ce même principe, Bin et al. [BQX10] préfèrent se focaliser sur la détection de la saisie des numéros de cartes bancaires au sein des pages webs. Leur système s'appuie sur une base de connaissances préalable des noms de banques, adresses IP et plages de numéros de cartes bancaires associés. Ainsi, dès lors qu'un utilisateur commence à saisir un numéro de carte bancaire dans une page web, le système en déduit le nom d'une banque. Il émet alors automatiquement une requête DNS inverse, afin de s'assurer que l'adresse IP du domaine actuellement visité est bien incluse dans les adresses IP connues pour la banque interrogée.

Un système d'analyse passive des informations DNS est également proposé par Bilge et al [BKKB11] afin de détecter les domaines contrefaits. Davantage ciblé sur la détection de sites de phishing, certains critères étudiés peuvent toutefois également s'appliquer à la détection du pharming. Par exemple, le nombre d'adresses IP retournées par une requête DNS sur le domaine visité, les résultats de requêtes reverse DNS, etc.

Enfin de manière plus générale, Ollman [Oll04] préconise d'étendre l'utilisation des moyens habituels de détection proposés pour le poste client (p.ex. pare-feu, anti-spyware).

4.2 Analyse et comparaison des pages webs : deux grandes catégories de méthodes pour la comparaison de documents HTML

Le Chapitre 3 a mis en évidence l'intérêt de l'analyse du code source HTML des pages webs, dans la différenciation des sites légitimes et contrefaits. Basé sur ce constat et notre recherche d'une technique de détection des sites webs contrefaits de pharming, nous nous sommes naturellement intéressés à l'analyse et la comparaison du contenu des pages webs.

Pour les raisons évoquées ultérieurement dans ce document (cf. section 5.2.2), nous avons choisi de focaliser notre comparaison de pages webs sur la comparaison de codes sources HTML. En effet, notre piste de recherche vise à confronter deux documents HTML afin de déterminer un degré de similitude/divergence des deux contenus étudiés. Dans notre étude, cette comparaison a pour vocation à être utilisée pour l'identification et/ou la différenciation de pages légitimes vs. des pages contrefaites.

Un document HTML (c.-à-d. le code source d'une page web) est avant tout un fichier texte, organisé selon une structure spécifique (cf. section 2.2.2.1). Deux grandes catégories de méthodes de comparaisons s'offrent alors : les méthodes de comparaison de textes qui travaillent sur le contenu du document de manière globale, ou les méthodes de comparaison qui se basent sur la structure du document étudié (nommée DOM).

Dans la suite de ce chapitre, nous nous intéressons donc à décrire ces deux grandes catégories de méthodes de comparaison, avec une attention particulière sur les méthodes de comparaisons de texte. Celles-ci se sont en effet révélées être mieux adaptées au contexte de notre étude (cf. explications détaillées en section 5.2.2.3).

4.2.1 Les méthodes de comparaison de textes

Les méthodes de comparaison de textes se décomposent entre différentes catégories/familles d'algorithmes (cf. figure 4.6) :

- Les algorithmes d'alignement de chaînes
- Les algorithmes de recherche de sous-chaînes
- Les algorithmes de mesure de similarité

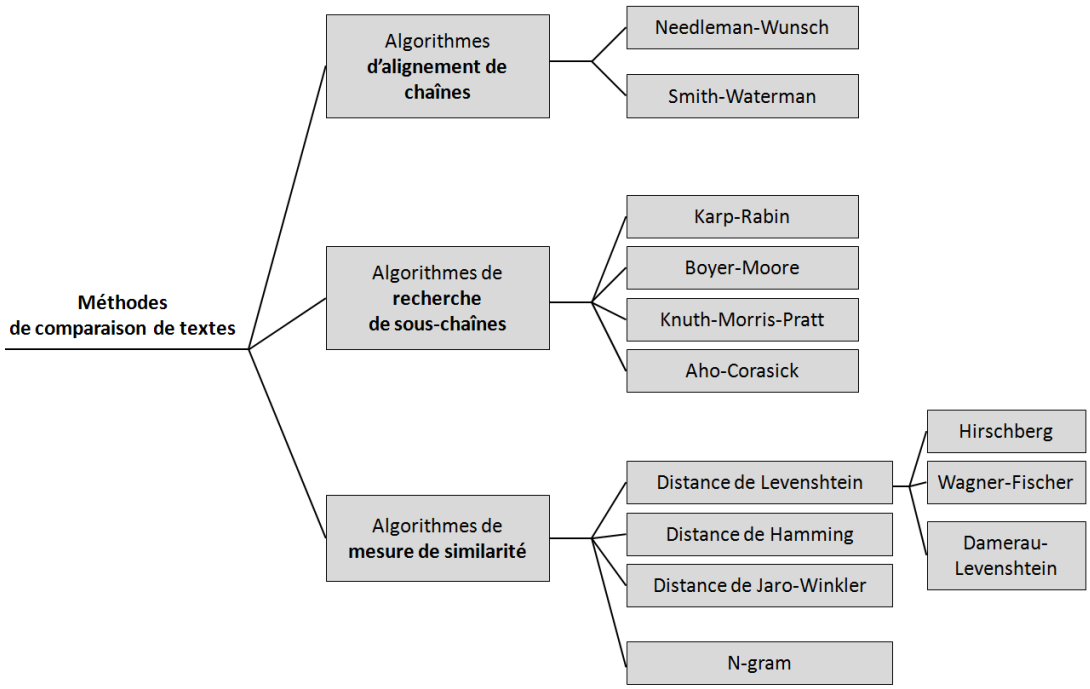

FIGURE 4.6 – Taxonomie des méthodes de comparaison de textes

Les algorithmes de comparaisons de textes étant particulièrement nombreux, chaque catégorie exposée ci-après ne sera illustrée qu'au travers du détail d'un à deux algorithmes maximum.

4.2.1.1 Les algorithmes d'alignement de chaînes

Cette famille d'algorithmes est typiquement utilisée en génétique, afin de comparer deux séquences (p.ex. ADN). Les algorithmes Needleman-Wunsch [NW70] et Smith-Waterman [SW81] en sont les exemples les plus connus. Ils font tous deux partie des techniques de programmation dynamique, dont le mode de fonctionnement vise à découper un problème en sous-problèmes, afin d'appliquer un traitement optimal à chacun d'entre-eux. La programmation dynamique s'appuie sur le principe que des sous-problèmes traités de façon optimale permettront d'aboutir à une solution optimale pour le problème global [Lik05].

Prenons l'exemple de l'**algorithme Needleman-Wunsch**, moins restrictif que l'algorithme Smith-Waterman. Il fonctionne sur le principe suivant : pour deux chaînes de caractères à comparer (x, y), on définit une matrice de similarité (à chaque colonne on associe un caractère de la chaîne x, et à chaque ligne on associe un caractère de la chaîne y). A cette matrice de similarité, on associe une pénalité de trou (p) ≤ 0. Cette dernière est utilisée pour initialiser les valeurs des premières ligne et colonne de la matrice (nommées *Coût*), selon la règle suivante :

$$\begin{aligned} Co\hat{u}t(i) &= Co\hat{u}t(i-1) + p \\ Co\hat{u}t(j) &= Co\hat{u}t(j-1) + p \end{aligned} \tag{4.1}$$

où *(i)* est le numéro de ligne et *(j)* est le numéro de colonne.

Le score de chaque cellule (C) de la matrice est ensuite déterminé selon les scores de ses cellules immédiatement adjacentes : (Cdiag(i-1,j-1)) pour cellule diagonale supérieure gauche, (Cup(i-1, j)) pour cellule supérieure et (Cleft(i,j-1)) pour cellule gauche, selon la formule suivante :

$$\text{Score C(i,j)} = max \begin{cases} \text{Score C}diag\text{(i-1,j-1)} + S(x(i), y(i)) \\ \text{Score C}up\text{(i-1,j)} + p \\ \text{Score C}left\text{(i,j-1)} + p \end{cases} \tag{4.2}$$

où *(i)* est le numéro de ligne, *(j)* est le numéro de colonne,
et S est la valeur de substitution des deux caractères étudiés selon une matrice pré-définie (p.ex. la matrice BLOSUM62 [oK]).

Puis, en partant de la fin de la matrice (c.-à-d. depuis la dernière cellule en bas à droite), on détermine le meilleur alignement (c.-à-d. qui permet d'aboutir à l'alignement d'un maximum de caractères communs) en remontant les cellules jusqu'au point d'origine de la matrice. Pour remonter les cellules, on choisit toujours la cellule de score le plus élevé parmi les 3 cellules immédiatement adjacentes (c.-à-d. cellules gauche, droite, ou diagonale supérieure gauche). A chaque déplacement latéral correspondra l'insertion d'un trou dans la chaîne y, tandis qu'à chaque déplacement vertical correspondra l'insertion d'un trou dans la chaîne x.

La figure 4.7 illustre un exemple de l'alignement des chaînes de caractères *CHAINE* et *CRAIE* en utilisant la matrice de substitution BLOSUM62 (pour plus de détails, cf. section 5.2.2.3.1), et une pénalité de trou à 0.

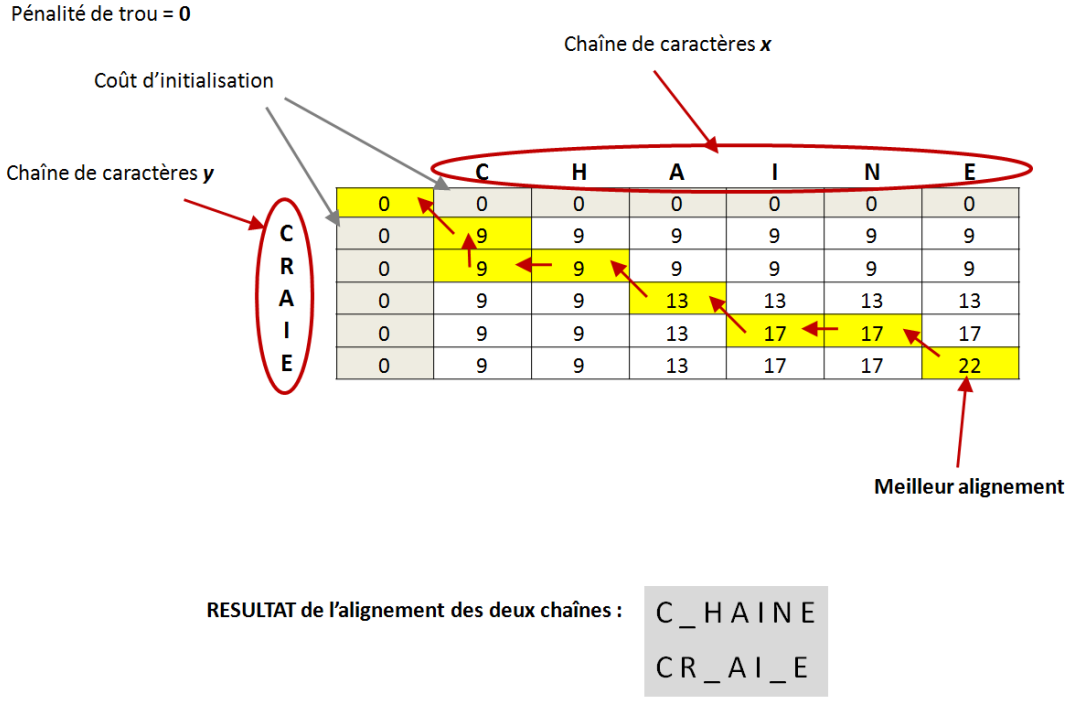

FIGURE 4.7 – Exemple de l'alignement de 2 chaînes de caractères, avec l'algorithme Needleman-Wunsch

L'**algorithme Smith-Waterman** présente un fonctionnement très similaire à l'algorithme Needleman-Wunsch. Néanmoins, il vise à s'appliquer plutôt sur des sous-parties d'une chaîne de caractères vs. une chaîne de caractères complète.

Le projet B.A.B.A (pour *Basic Algorithms of Bioinformatics Applet*) [Sou]) est un exemple d'implémentation de ces deux algorithmes. Il permet de tester en ligne les deux algorithmes d'alignement de chaînes énoncés dans cette section.

4.2.1.2 Les algorithmes de recherche de sous-chaînes

Les algorithmes de recherche de sous-chaînes ont pour but de rechercher une chaîne de caractères à l'intérieur d'une autre chaîne de caractères, dite de référence. Bien connues dans le domaine de la sécurité informatique, les attaques de type *brute force* en sont une application typique : par le test successif de toutes les combinaisons possibles, elles visent à retrouver un mot de passe, une clé de chiffrement, etc.

Les algorithmes de recherche de sous-chaînes sont particulièrement nombreux [CL97]. Parmi les plus connus, on peut citer les algorithmes Karp-Rabin [KR87], Boyer-Moore [BM77], Knuth-Morris-Pratt [KMP77], Aho-Corasick [AC75], etc. Ils traitent la chaîne de caractères à trouver de gauche à droite, ou de droite à gauche, en s'appuyant sur des dictionnaires, des fonctions de hachage (représentative de la base de caractères possibles), etc.

Prenons l'exemple de l'**algorithme Boyer-Moore**, considéré comme l'un des plus efficaces de sa catégorie. Il a la particularité de traiter les chaînes de caractères de droite à gauche. Si l'on considère un texte de référence (nommé T) constitué de 25 caractères, au sein duquel on doit rechercher une chaîne de 5 caractères (nommée C, dont la longueur exprimée en caractères est nommée X), l'algorithme se déroule de la manière suivante : si la dernière lettre de C ne correspond pas à la dernière lettre des X premiers caractères de T, on fait un saut de X caractères dans T. On regarde alors la correspondance entre la dernière lettre de C et la dernière lettre du mot actuellement étudié dans T. Si cela ne correspond pas, on fait un nouveau saut de X caractères dans T, et ainsi de suite jusqu'à atteindre la fin de T. Dès lors qu'on obtient une correspondance d'un dernier caractère (entre C et le mot actuellement étudié dans T), on remonte au caractère précédent et ainsi de suite. Dès qu'on détecte une non-concordance au sein des deux mots étudiés, on passe au mot suivant de T.

A noter que la taille du saut à effectuer dans le texte de référence (indiquée par X dans le déroulement simplifié ci-dessus) est en réalité déterminée à l'issue d'une phase de pré-apprentissage de l'algorithme, effectuée sur les deux chaînes à comparer.

La figure 4.8 illustre un exemple de résultat obtenu avec l'algorithme Boyer-Moore. Elle a été établie à partir de l'implémentation proposée par Charras [CL97].

4.2.1.3 Les algorithmes de mesure de similarité

Les algorithmes de mesure de similarité ont pour but de calculer la distance entre deux éléments de même type (typiquement deux chaînes de caractères, deux documents, etc.). Cette distance est représentative du nombre d'opérations minimum nécessaires pour passer d'une chaîne de caractères à l'autre.

Les principaux algorithmes de mesure de similarité sont : la Distance de Levenshtein [Lev66] et ses dérivés (p.ex. Wagner-Fischer [WF74], Hirschberg [Hir75], Damerau [Dam64]-Levenshtein) - plus connus sous la terminologie de Distance d'édition -, la Distance de Hamming [Ham50] ou encore la Distance de Jaro-Winkler [Win06].

Prenons l'exemple de l'**algorithme de Distance de Levenshtein**, qui adresse davantage d'opérations que ses pairs. Il fait partie de la famille des techniques de programmation dites dynamiques (cf. section 4.2.1.1). Cet algorithme prend en compte trois types d'opérations : l'ajout, la suppression et la modification (également appelée substitution) de texte.

La base utilisée est similaire à celle vue pour l'algorithme Needleman-Wunsch (cf. section 4.2.1.1), à savoir la définition d'une matrice de similarité. Soit deux chaînes de caractères à comparer x, y. A chaque colonne, on associe un caractère de la chaîne x, et à chaque ligne on associe un caractère de

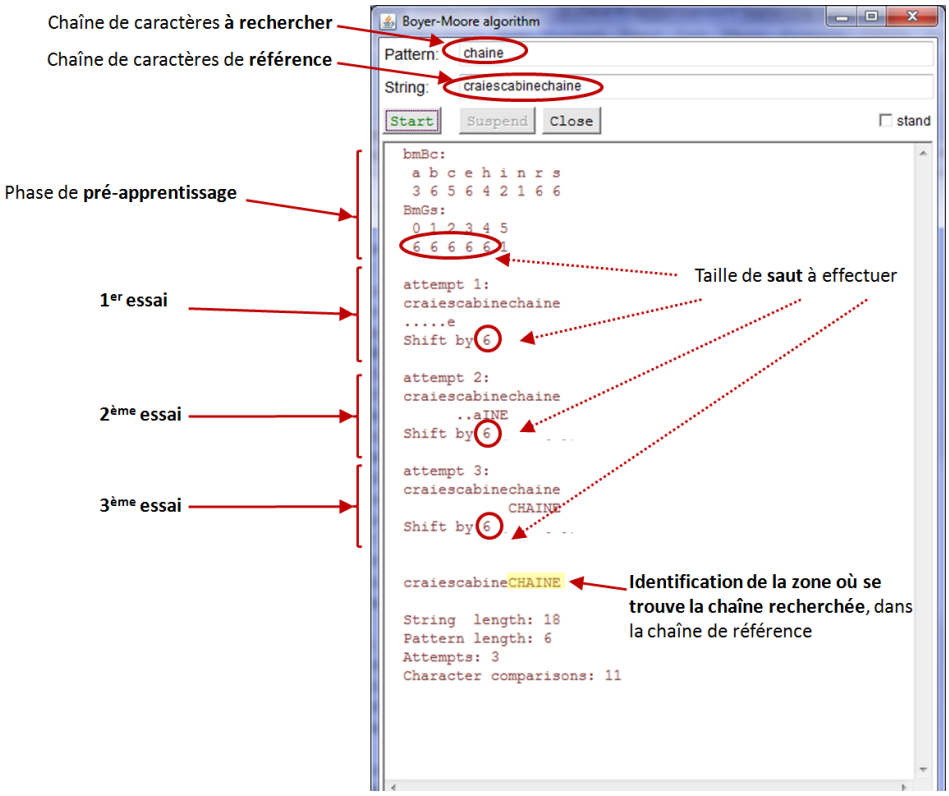

FIGURE 4.8 – Exemple de recherche de sous-chaîne, établi avec l'implémentation [CL97] de l'algorithme Boyer-Moore

la chaîne *y*. Puis, on définit les 3 pénalités choisies pour l'ajout *Pajout*, la modification *Pmodif* et la suppression *Psuppr*. La matrice de coût est ensuite initialisée avec les règles suivantes :

- Le coût d'ajout est indiqué sur la 1ère ligne horizontale. Il est établi à partir de *Pajout* (p.ex. si *Pajout*=1, le coût d'ajout correspondant est : 1,2,3,4 etc.)
- Le coût de suppression est indiqué sur la 1ère ligne verticale. Il est établi à partir de *Psuppr* (p.ex. si *Psuppr*=1, le coût de suppression correspondant est : 1,2,3,4 etc.)
- Chaque cellule restante est initialisée en fonction de la correspondance des lettres de sa colonne et de sa ligne. Cette valeur ainsi attribuée est appelée : Coût de modification (*Coût modif*) de la cellule. S'il y a correspondance entre les lettres, le coût de modification est de 0. Sinon il est égal à *Pmodif*.

Le score de chaque cellule (C) de la matrice est ensuite déterminé selon les scores de ses cellules immédiatement adjacentes : (Cdiag(i-1,j-1)) pour cellule diagonale supérieure gauche, (Cup(i-1, j)) pour cellule supérieure et (Cleft(i,j-1)) pour cellule gauche, selon la formule suivante :

$$\text{Score C(i,j)} = min \begin{cases} \text{Score C}diag\text{(i-1,j-1)} + \text{Coût modif(i-1,j-1)} \\ \text{Score C}up\text{(i-1,j)} + Psuppr \\ \text{Score C}left\text{(i,j-1)} + Pajout \end{cases} \tag{4.3}$$

où *(i)* est le numéro de ligne et *(j)* est le numéro de colonne.

Puis, en partant de la fin de la matrice (c.-à-d. depuis la dernière cellule en bas à droite), on détermine le chemin des opérations (ajouts, suppressions, modifications) en remontant les cellules jusqu'au point d'origine de la matrice. Pour remonter les cellules, on choisit toujours la cellule de score le plus faible parmi les 3 cellules immédiatement adjacentes (c.-à-d. cellules gauche, droite, ou diagonale supérieure gauche). A chaque changement de valeur au cours du chemin des opérations, correspond l'ajout, la modification ou la suppression d'un caractère.

La figure 4.9 illustre un exemple de calcul de Distance de Levenshtein pour les mots *CHAINE* et *CRAIE*, réalisé à partir de l'implémentation de Kleiweg [Kle], avec des pénalités d'opération établies à 1.

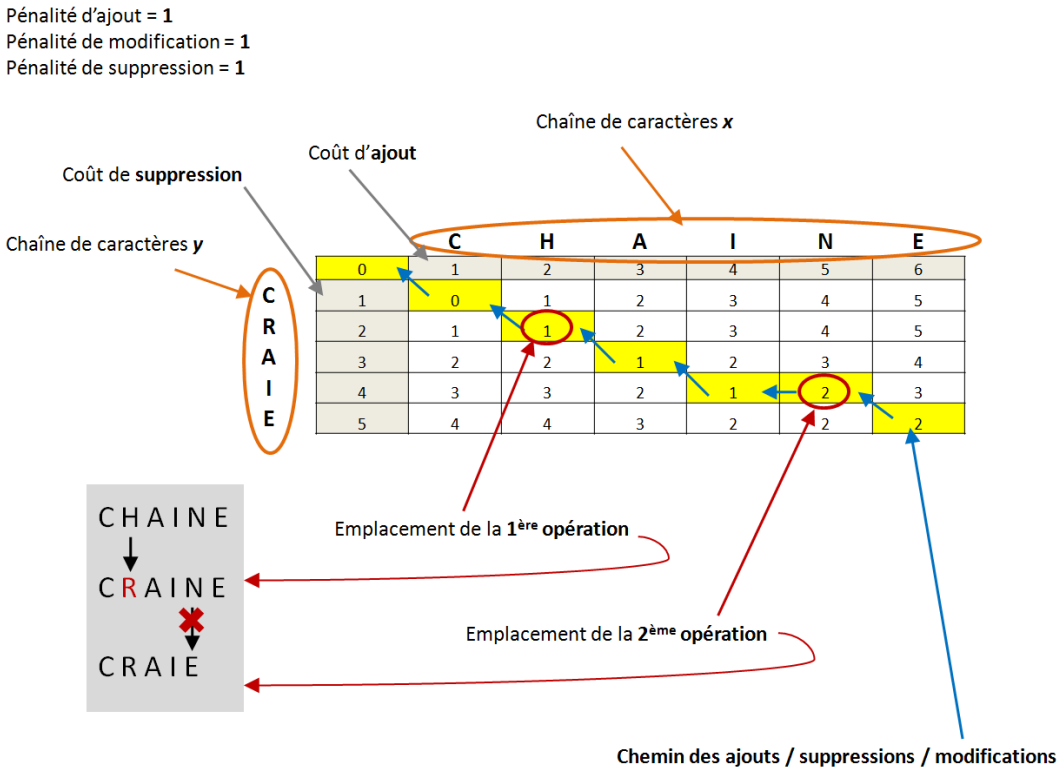

FIGURE 4.9 – Exemple de calcul de Distance de Levenshtein sur 2 chaînes de caractères, avec l'implémentation de Kleiweg [Kle]

Un cas particulier du calcul de la Distance d'édition consiste à déterminer la plus longue sous-séquence commune (plus connue sous la terminologie *LCS - Longest Common Subsequence - problem* en anglais) à deux chaînes de caractères. Celle-ci est obtenue en établissant une matrice des deux chaînes à comparer *x, y* dans laquelle les pénalités d'ajout et de suppression sont fixées à 1, tandis que le coût de suppression est établi à 0. Autrement dit, on ne prend en compte que les ajouts ou suppressions de caractères, et les modifications de caractères sont ignorées [BHR00].

Ce cas particulier du calcul de la Distance d'édition a servi de base à l'élaboration du programme *diff* [Mye86] [WMMM90] utilisé, sous Unix, pour la comparaison de textes.

L'algorithme de **Distance de Hamming** est fortement utilisé par les codes correcteurs d'erreurs [Sym10]. Il ne s'intéresse qu'à la substitution de caractères. A ce titre, l'algorithme ne peut être appliqué que sur deux chaînes de longueur identique. Le résultat qu'il délivre correspond au nombre de caractères modifiés. Par exemple, la Distance de Hamming entre les chaînes "phishing" et "pharming" est de 3.

L'algorithme de **Distance de Jaro-Winkler** est utilisé de préférence sur des chaînes de caractères de faible longueur [Win06], telles que les mots de passe. Le score délivré, compris entre 0 et 1, est représentatif du degré de similarité des deux chaînes comparées. Plus il est élevé, plus les chaînes sont ressemblantes. Dans son calcul de distance, cet algorithme ne prend en compte que les caractères identiques et déplacés. Son déroulement s'effectue en trois étapes :

1. Au sens de l'algorithme Jaro-Winkler, les deux chaînes à comparer sont considérées comme ressemblantes si le nombre de caractères transposés (transp) est inférieur ou égal à :

$$\left(\frac{max(|c1|,|c2|)}{2}\right)-1 \tag{4.4}$$

où *c1* et *c2* représentent le nombre de caractères des deux chaînes à comparer

Dans ce cas, la variable (t) - utilisée ci après pour calculer la distance de Jaro - prend la valeur de $(\frac{transp}{2})$. Sinon (t) = 0.

2. La distance de Jaro est alors calculée, selon la formule suivante :

$$\text{Distance de Jaro} = \frac{1}{3}\left(\frac{m}{|c1|} + \frac{m}{|c2|} + \frac{m-t}{m}\right) \tag{4.5}$$

où m est le nombre de caractères identiques (et au même emplacement) entre les 2 chaînes comparées,
où $c1$ et $c2$ représentent le nombre de caractères des deux chaînes à comparer,
et t est représentatif du nombre de caractères transposés/déplacés. Il est calculé selon la méthode expliquée ci-dessus.

3. Enfin, la distance de Winkler est calculée ainsi :

$$\text{Distance de Winkler} = \text{Distance de Jaro} + \left(l \cdot p(1 - \text{Distance de Jaro})\right) \tag{4.6}$$

où l est le nombre de caractères identiques entre les 2 chaînes comparées (sa valeur maximale est 4).
et p est un coefficient ayant pour but de favoriser les chaînes avec un préfixe commun. Sa valeur recommandée est 0.1.

Dans la catégorie des algorithmes de mesure de similarité, on peut également parler des approches de type ***N-gram***. Celles-ci sont généralement utilisées en traitement du signal pour corriger des erreurs de transmission (p.ex. l'algorithme de Viterbi), ou en traitement automatique du langage naturel pour effectuer de la reconnaissance vocale, de la reconnaissance de textes, etc.

Une étude menée par Cavnar [CT94] a notamment démontré l'efficacité de ce type d'approche pour la classification de texte/document. A un *N-gram* correspond un ensemble de N caractères consécutifs. Le principe de fonctionnement simplifié est alors le suivant :

1. On démarre par une phase de pré-apprentissage, c.-à-d. on prend des documents issus des différentes catégories de classifications souhaitées. Pour chacun de ces documents dits de référence, on détermine l'ensemble des *N-gram* contenus ainsi que leurs fréquences d'apparition. Ainsi, on peut en déduire un profil type associé à chaque catégorie de classification.
2. Chaque document à classifier est ensuite analysé selon le même principe, afin de déterminer les *N-gram* qu'il contient et leurs fréquences d'apparition.
3. Reste alors à effectuer un calcul de distance entre le document à référencer et les profils types pré-établis, la distance la plus courte permettant de définir la catégorie du document.

La figure 4.10 illustre un exemple de classification d'un document basée sur une approche *N-gram*, à partir de profils types créés.

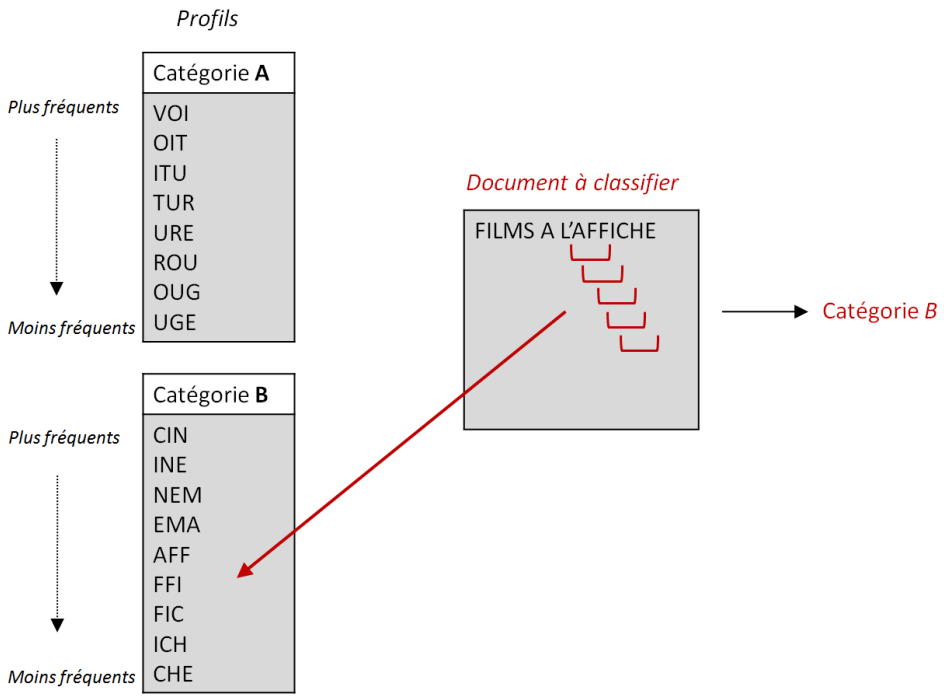

FIGURE 4.10 – Exemple de classification d'un document, à partir d'une approche *N-gram* à base de triplets

4.2.2 Les méthodes de comparaison de structures

D'autres méthodes de comparaison peuvent s'appliquer aux codes sources HTML. En effet de par leur contenu, ces fichiers texte peuvent également être examinés en fonction de leur structure bien particulière et ordonnée (c.-à-d. via les balises).

La différence essentielle avec les méthodes de comparaison de textes vues en section 4.2.1, réside dans le fait que cette fois-ci la comparaison se focalise sur la structure des documents étudiés. En effet, les algorithmes détaillés précédemment font abstraction de la structure, qu'ils considèrent comme du texte au même titre que le contenu réel de la page.

Ces méthodes de comparaison de structures sont dites en arbres (on parle notamment de structure DOM[1]). Elles s'appliquent sur des documents de même type présentant une arborescence ordonnée. On y distingue deux familles d'algorithmes (cf. figure 4.11) : les algorithmes d'alignement et les algorithmes de mesure de similarité.

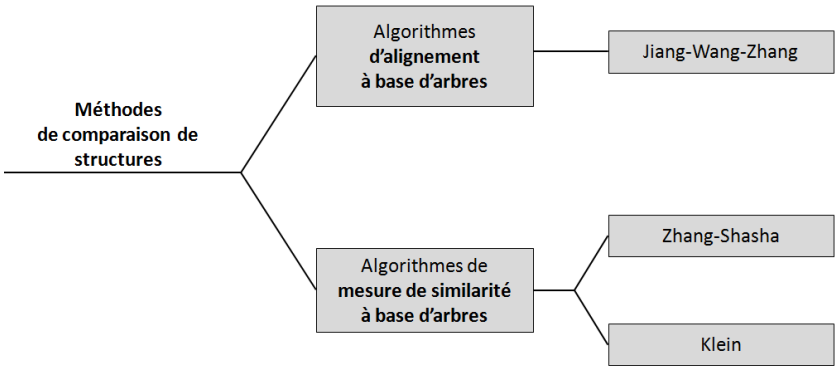

FIGURE 4.11 – Taxonomie des méthodes de comparaison de structures

Les bases communes à ces deux familles d'algorithmes sont les suivantes (cf. figure 4.12) : chaque document à comparer est vu comme un ensemble de nœuds (p.ex. les balises), reliés les uns aux autres selon un ordre déterminé. L'ensemble constitué étant nommé arbre complet et noté T. Le nœud d'origine où chaque nœud intermédiaire est vu comme la racine r d'un arbre F constitué des nœuds inférieurs qui lui sont raccordés. La notation de chaque sous-arbre (compris d'une racine et de l'arbre inférieur qui lui est associé) au sein de l'arbre complet est la suivante : $r(F)$.

A l'image des algorithmes présentés en section 4.2.1, les méthodes de comparaison de structures détaillées ici s'appuient sur l'établissement de matrices de similarité. Au sein de ces matrices, les colonnes et les lignes symbolisent les nœuds des arbres complets à comparer.

4.2.2.1 Les algorithmes d'alignement

Comme nous l'avons vu en section 4.2.1.1, cette famille d'algorithmes est typiquement utilisée en génétique, afin de comparer deux séquences (p.ex. ADN). L'algorithme Jiang et al. [JWZ94] est l'exemple le plus connu des algorithmes d'alignement basés sur les arbres.

Le principe général est le suivant : deux séquences à comparer, représentées par des arbres complets, sont alignées grâce à l'application des opérations d'ajout, suivies par les opérations de suppression. Là encore, le calcul du meilleur alignement est effectué grâce à l'utilisation d'une matrice de substitution (pour plus de détails sur la notion de matrice de substitution, cf. section 4.2.1.1).

La figure 4.13 - extraite de la présentation de Touzet [Tou05] - illustre les opérations réalisées pour passer de l'*arbre complet* T à l'*arbre complet* T' : on commence par ajouter les nouveaux nœuds f et d dans l'arbre de droite. Puis, on supprime les nœuds b et d dans l'arbre de gauche.

1. cf. section 2.2.2.1 pour plus de détails.

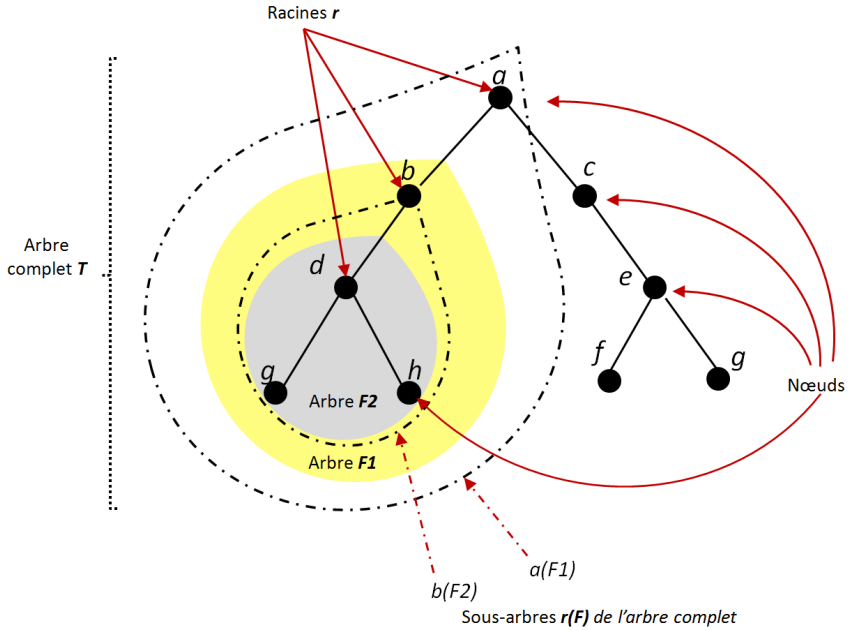

FIGURE 4.12 – Vue simplifiée des composants d'un arbre

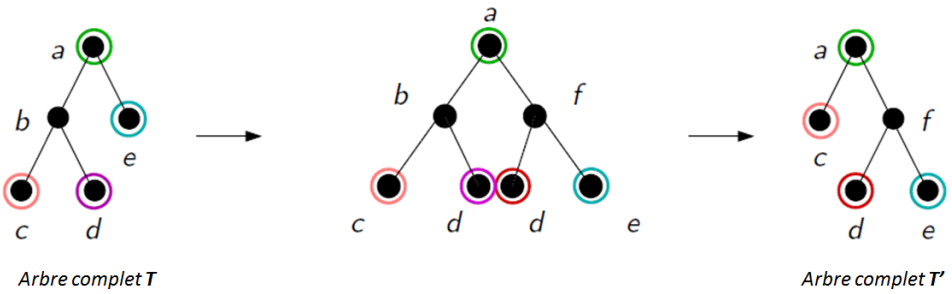

FIGURE 4.13 – Exemple d'opérations réalisées pour l'alignement de deux arbres complets T et T' [Tou05]

4.2.2.2 Les algorithmes de mesure de similarité

Les algorithmes de mesure de similarité basés sur les arbres ont le même but que ceux vus dans les méthodes de comparaison de texte (cf. section 4.2.1.3), à savoir : calculer une distance dite d'édition, représentative du nombre d'opérations minimum nécessaires (ajout, suppression ou modification) pour passer d'un document à l'autre.

Les algorithmes les plus connus dans ce domaine sont ceux de Zhang-Shasha [ZS89] et Klein [Kle98].

Le calcul de la Distance d'édition basée sur les arbres se décompose en deux temps principaux :

1. La distance de chaque sous-arbre est déterminée selon le calcul suivant :

$$Distance\Big(r(F), r'(F')\Big) = min\begin{cases} Distance\Big(F, r'(F')\Big) + Csuppr(r) \\ Distance\Big(r(F), F'\Big) + Cajout(r') \\ Distance\Big(F, F'\Big) + Cmodif(r, r') \end{cases} \tag{4.7}$$

où *Csuppr*, *Cajout* et *Cmodif* sont respectivement les coûts de suppression, ajout et modification d'un noeud.

2. Puis la distance d'édition de l'arbre complet est déterminée suivant deux approches : gauche ou droite. A titre d'exemple, l'algorithme Zhang-Shasha utilise la décomposition gauche, tandis que l'algorithme Klein peut utiliser l'une ou l'autre selon la structure de l'arbre. Nous détaillons ci-après la version gauche du calcul de la distance d'édition de l'arbre complet (La version droite et de plus amples détails sur les méthodes de comparaison à base d'arbres sont disponibles en [DT03] et [Bil05]).

$$Distance\Big(r(F)\circ T, r'(F')\circ T'\Big) = min\begin{cases} Distance\Big(F\circ T, r'(F')\circ T'\Big)+Csuppr(r) \\ Distance\Big(r(F)\circ T, F'\circ T'\Big)+Cajout(r') \\ Distance\Big(r(F), r(F')\Big)+Distance\Big(T, T'\Big) \end{cases} \tag{4.8}$$

où $\circ$ symbolise le rattachement d'un arbre (ou sous-arbre) à son arbre complet.

Au final, la matrice de similarité établie permet d'indiquer l'ensemble des opérations à effectuer pour passer d'un arbre complet à l'autre. Les ajouts de nœuds sont symbolisés par des tracés verticaux, tandis que les suppressions de nœuds sont symbolisées par des tracés horizontaux. Les tracés en diagonale indiquent une modification de noeud ou sa conservation.

La figure 4.14 - extraite de la présentation de Touzet [Tou05] - illustre les opérations menées pour passer de l'*arbre complet T* à l'*arbre complet T'* : on supprime le nœud *b*, puis on ajoute le nœud *f* en amont des nœuds *d* et *e*.

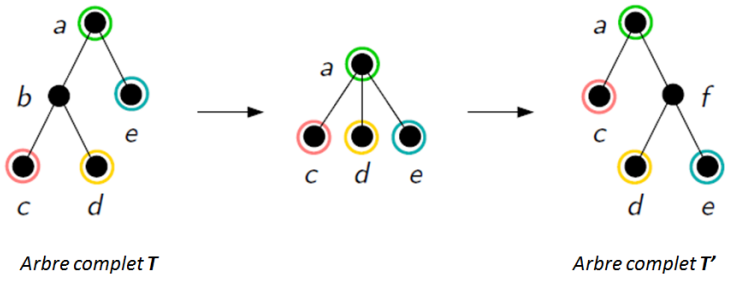

FIGURE 4.14 – Exemple d'opérations considérées par le calcul de Distance d'édition pour passer de l'arbre complet T à l'arbre complet T' [Tou05]

4.2.3 Application des méthodes de comparaison aux pages webs

Dans cette section, nous discutons exclusivement des algorithmes qui se prêtent à la comparaison de pages webs, aux travers de travaux existants.

4.2.3.1 Les méthodes de comparaison de textes

Les algorithmes de recherche de sous-chaînes sont typiquement utilisés par les moteurs de recherches et autres robots qui scrutent les pages webs en vue de les référencer, les classifier, etc. [GJ09] [ACC+09].

Les algorithmes de mesure de similarité sont quant à eux couramment utilisés pour la comparaison de pages webs, tant pour la mise en exergue des zones de changements dans le cadre de l'administration d'un site web, que pour la détection d'éventuels sites frauduleux. On peut citer par exemples les travaux de Medvet et al. [MKK08] qui font appel à la Distance de Levenshtein pour comparer deux pages webs (ou plus exactement des portions de texte contenues dans celles-ci), afin de détecter d'éventuelles contrefaçons. De leur côté, Reddy et al. [RRJ11] appuient leur détection des sites contrefaits sur l'utilisation d'une liste blanche d'URLs/adresses IP associées, afin de déterminer la légitimité du site visité par l'utilisateur. Leur concept fait appel à la Distance de Levenshtein pour comparer les URLs. Fu et al. [FDWL06] proposent quant à eux d'utiliser la Distance de Levenshtein pour détecter les attaques Unicode. Celles-ci sont typiquement utilisées dans des URLs afin de passer des commandes visant à

corrompre la machine de l'utilisateur. De leur côté, Simon et al. [SL05] utilisent l'algorithme Jaro-Winkler pour améliorer les extractions automatisées de contenus webs.

Enfin, l'approche N-gram est couramment utilisée sur le web [WTV+10] [AS] [LCJ+10] aussi bien pour constituer des corpus de mots, qui ont vocation à être utilisés par des outils qui doivent réaliser un découpage de mots/phrases intelligent (p.ex. pour appliquer les césures au bon endroit), que pour permettre la comparaison de documents (p.ex. des pages webs).

4.2.3.2 Les méthodes de comparaison de structures

Seuls les algorithmes de mesure de similarité basés sur les arbres se prêtent à la comparaison de pages webs (vs. les algorithmes d'alignement basés sur les arbres).

Dans leur étude, Mikhael et al. [MS05] ont implémenté un outil de comparaison des pages HTML - nommé *VDiff* -, qui se base sur l'algorithme Zhang-Shasha. Leur outil permet de détecter les changements de structures, c.-à-d. les changements de balises (ajout, suppression, modification). Son usage est typiquement destiné aux webmasters qui veulent apprécier les changements apportés aux pages webs par les différents contributeurs, avant publication.

De leur côté, Kim et al. [KPKC07] attribuent des poids différents aux balises du document HTML (c.-à-d. aux nœuds des arbres), selon la taille d'affichage des informations dans le navigateur web. L'idée est de faciliter la collecte/recherche des informations essentielles contenues dans les pages webs, via une comparaison de contenus effectuée grâce à des algorithmes de Distance d'édition basés sur les arbres.

Précisons que le postulat de départ de ces deux études est de considérer que la page HTML est écrite selon la structure XML .

Les travaux de Garofalakis et al. [GK05] et ceux de Li et al. [LLL+07] s'appuient également sur l'algorithme Zhang-Shasha pour la comparaison de documents XML.

4.3 Synthèse du chapitre

Ce chapitre a présenté le pharming - qui associe corruption DNS et site web frauduleux - ainsi que les différentes zones de vulnérabilité et/ou cibles d'attaques, depuis le réseau client jusqu'au site web visité.

Il est aujourd'hui de notoriété publique que le manque de protection du protocole DNS - première cible de ces attaques - ne peut éviter d'éventuelles compromissions des informations DNS légitimes, ce qui facilite l'exposition de l'Internaute à des sites webs frauduleux sans signe apparent de l'attaque perpétrée. Des efforts notables sont actuellement en cours de déploiement côté FAI/serveur web pour pallier cette lacune, via l'implémentation des extensions de sécurité DNSSEC. Néanmoins, nous avons vu que le réseau client demeure une cible privilégiée et insuffisamment protégée.

Côté client, l'ensemble des propositions de détection/prévention du pharming se rejoignent sur une volonté de mieux contrôler le contenu des pages webs visitées. Parmi les différentes pistes d'analyses de contenu des pages webs, nous avons choisi de détailler les méthodes de comparaison pouvant s'appliquer à l'étude des codes sources HTML (au niveau texte et structure), amorçant ainsi un des vecteurs de détection utilisé dans notre contribution.